BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

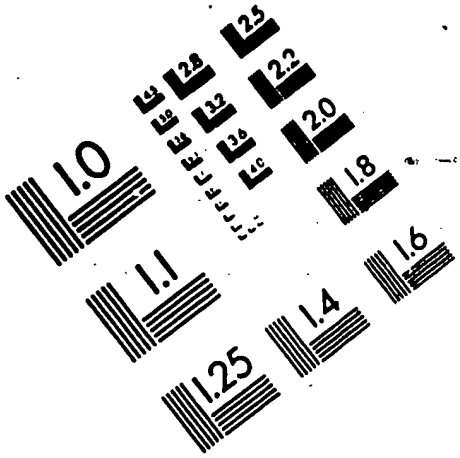
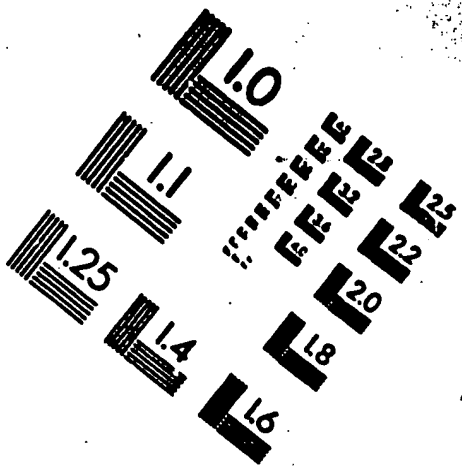
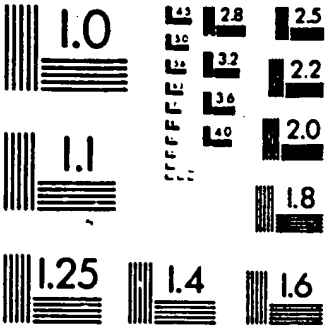
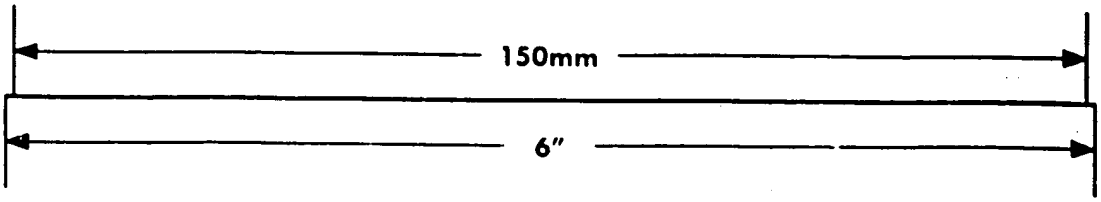
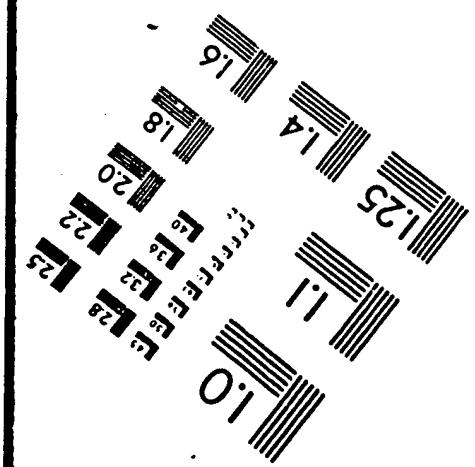


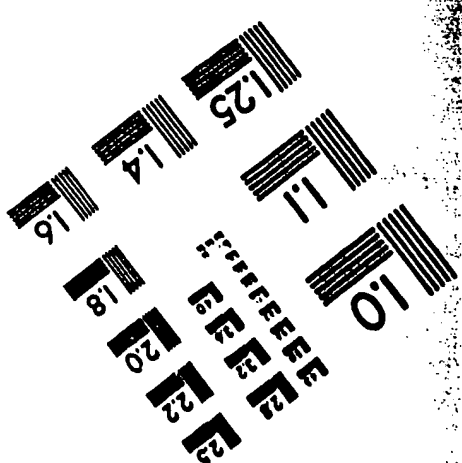
IMAGE EVALUATION TEST TARGET (MT-3)











PHOTOGRAPHIC SCIENCES CORPORATION
770 BASKET ROAD
P.O. BOX 338
WEBSTER, NEW YORK 14580
(716) 265-1600

microunity

Zeus System Architecture

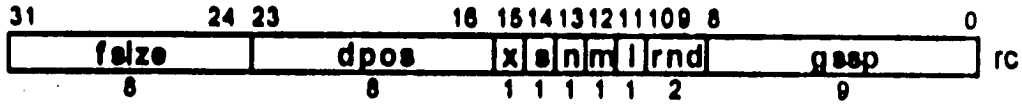
MicroUnity

Craig Hansen
Chief Architect

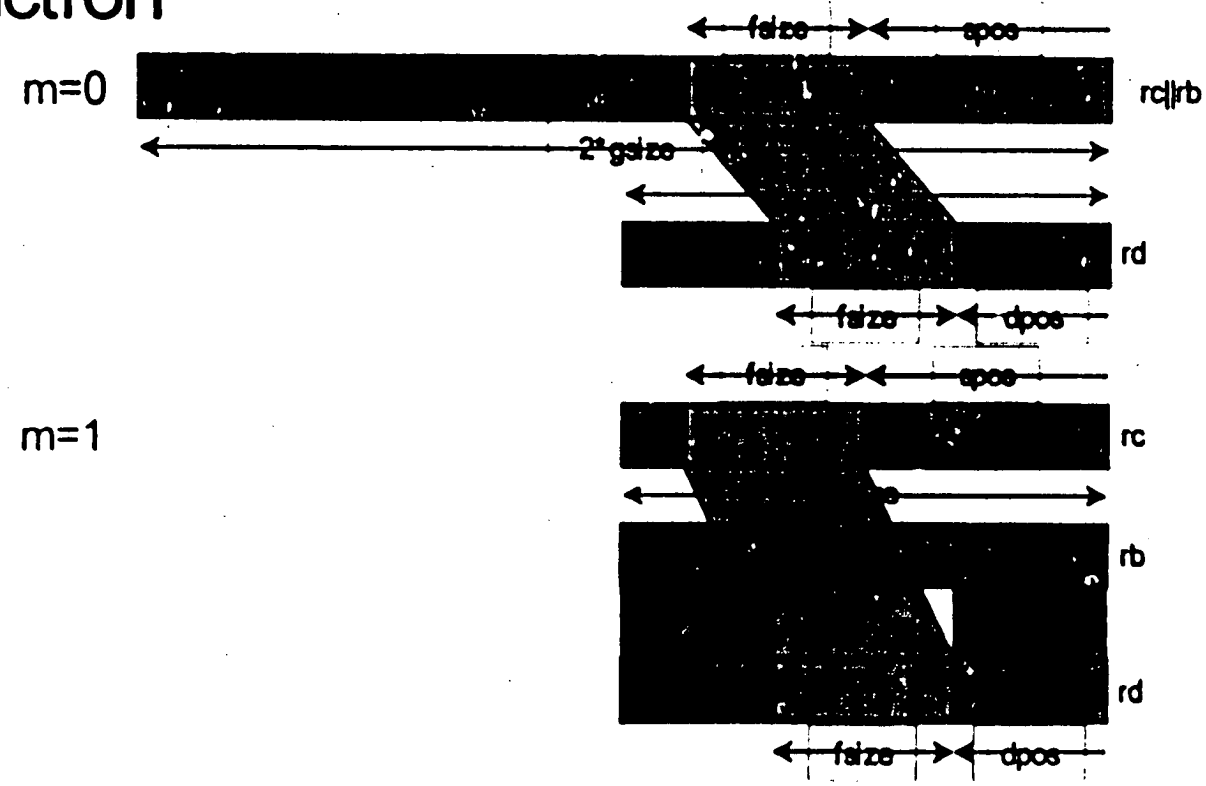
MicroUnity Systems Engineering, Inc.
475 Potrero Avenue
Sunnyvale, CA 94086.4118
Phone: 408.734.8100
Fax: 408.734.8136
email: craig@microunity.com
http://www.microunity.com

Crossbar extract control

- layout

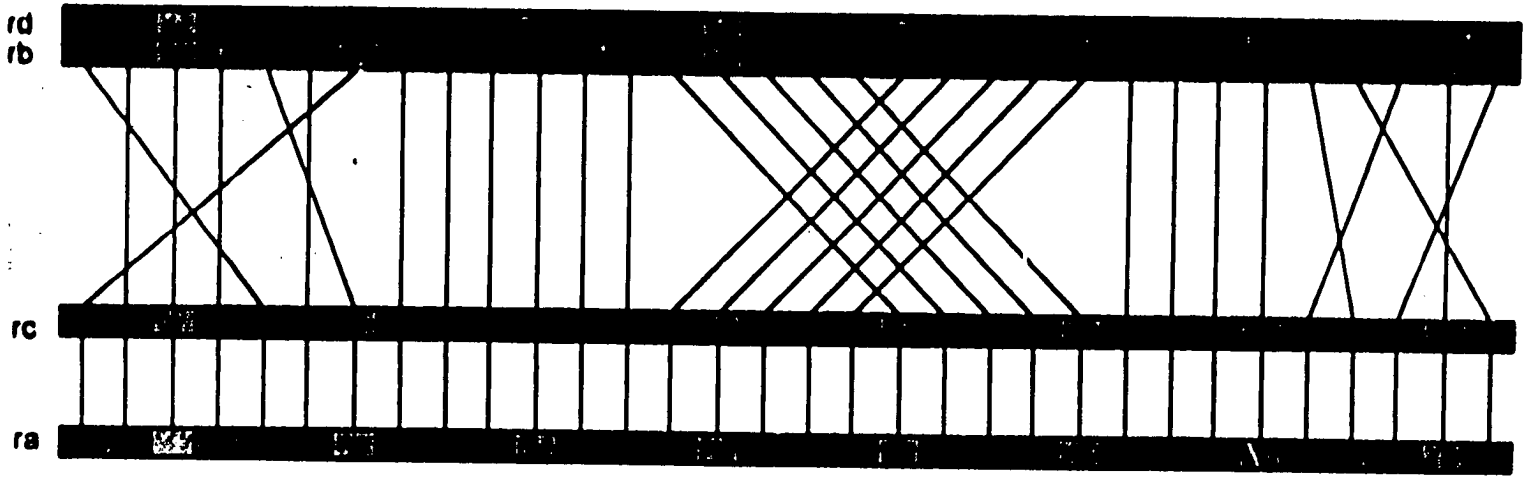


- function



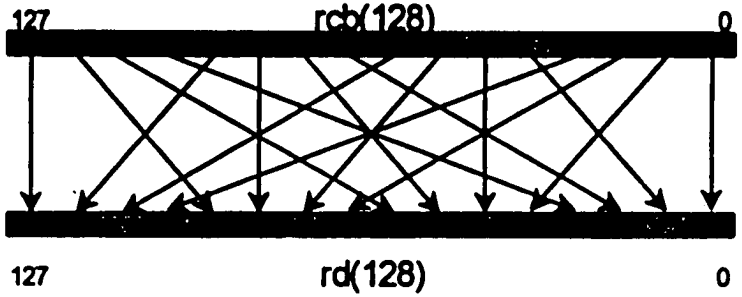
Crossbar Select bytes

- X.SELECT.8 ra=rc,rd,rb



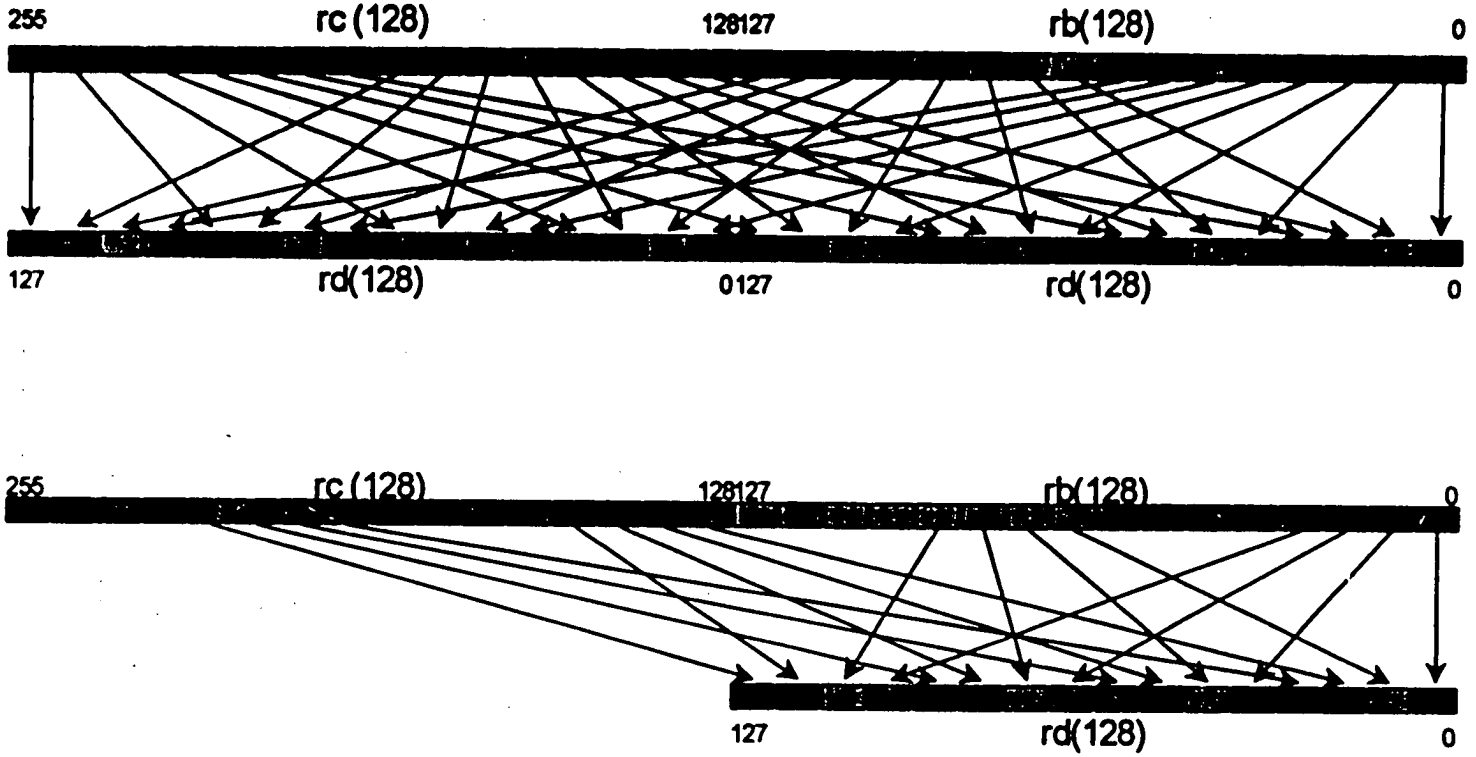
4-way shuffle bytes within hexlet

- XSHUFFLEI.128 rd=rcb,8,4



4-way shuffle bytes within triclet

- XSHUFFLEI.128 rd=rc,rb,8,4

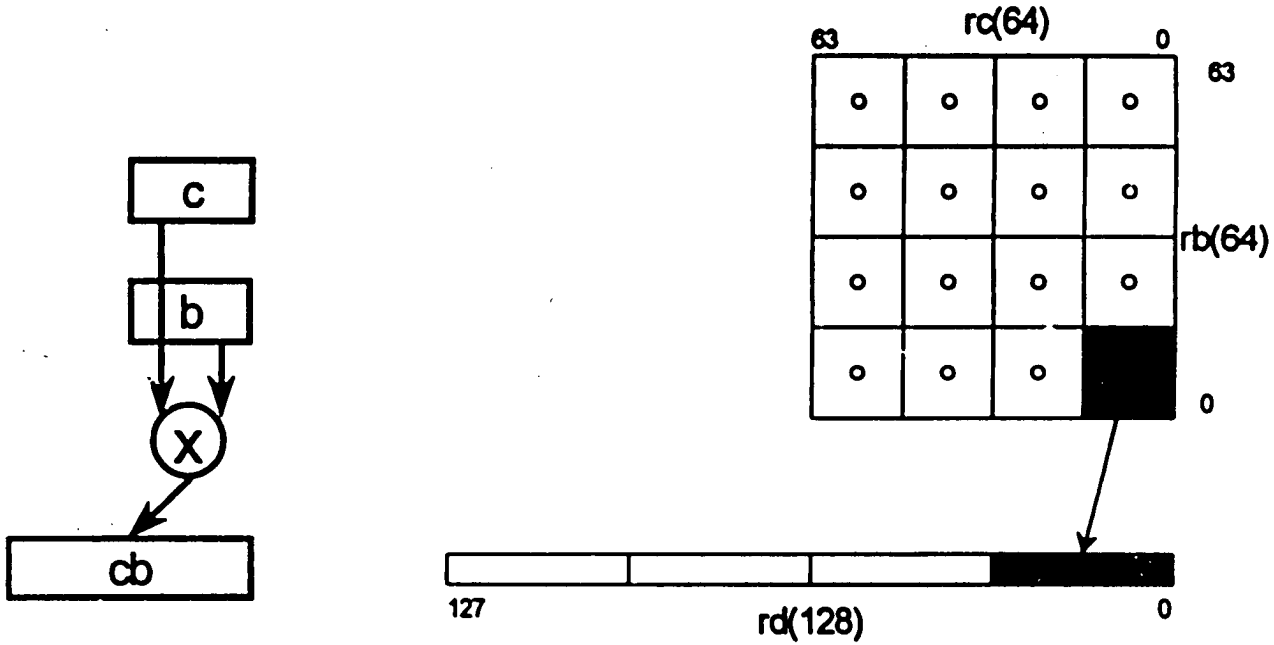


Ensemble Instructions

- Multiply
  - Fixed-point
    - size-doubling
    - extract
  - Floating-point
  - Complex
  - Polynomial
  - Galois Field
  - Convolve
  - Multiply-add
  - Scale-add
  - Multiply-sum
- Floating-point
  - Add, Subtract, Divide, Sum
  - Inflate, Deflate, Float, Sink
  - Reciprocal Estimate
  - Reciprocal Square Root Estimate
- Fixed-point
  - Sum
  - Log-most

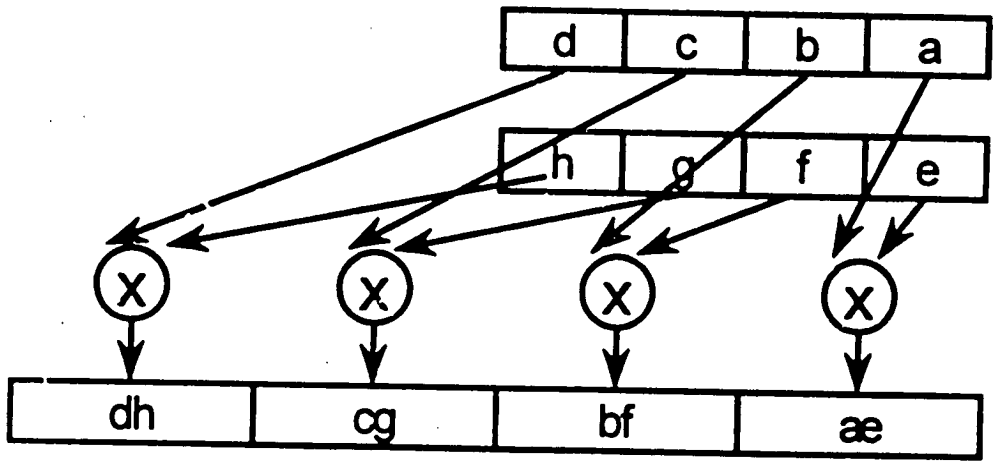
Multiply

- $rd_{32} = rc_{16} * rb_{16}$



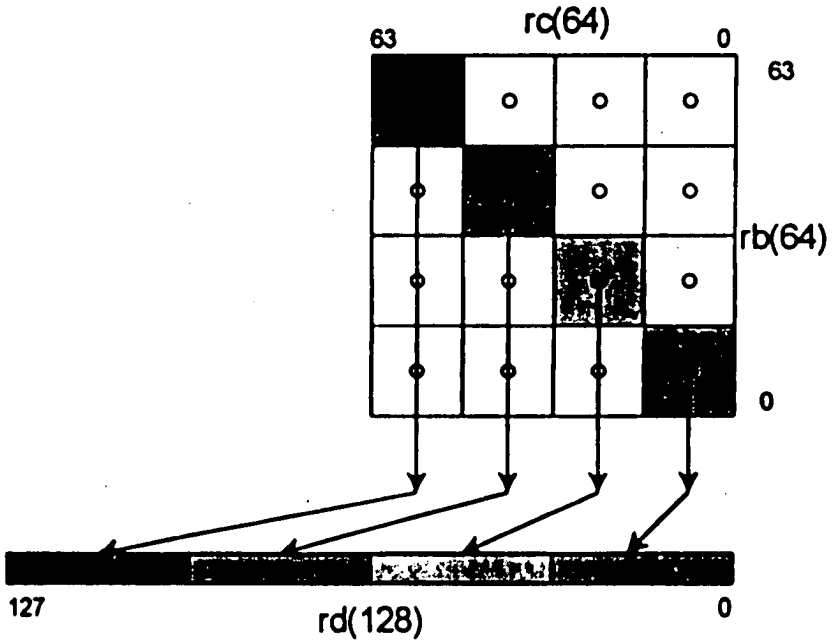
Ensemble multiply

- $rd_{128} = rc_{64} * rb_{64}$



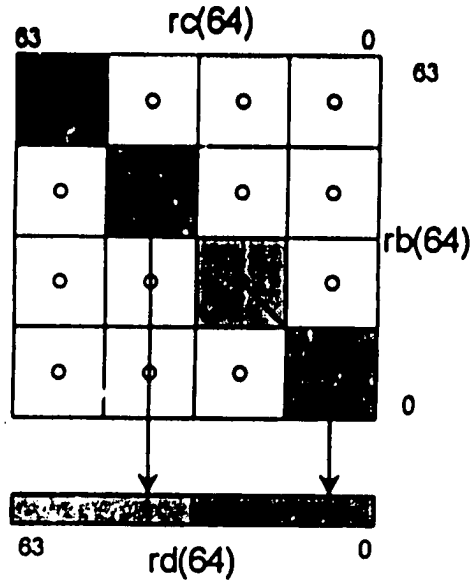
Ensemble multiply

- $rd_{128} = rc_{64} * rb_{64}$



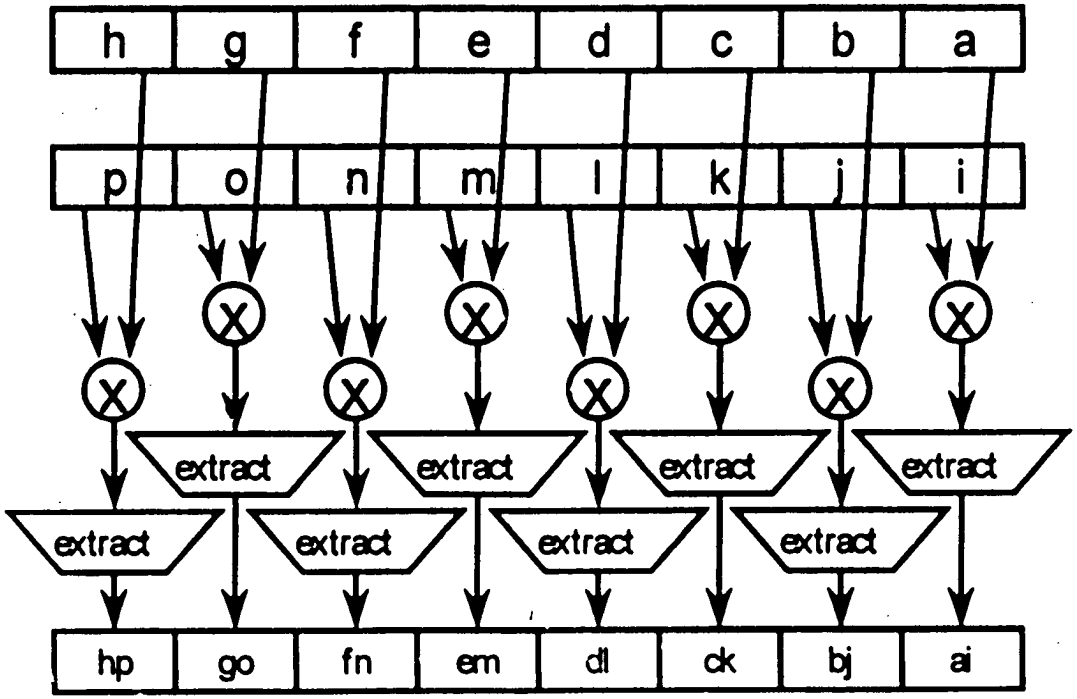
MMX PMADDWD

- $rd_{128} = rc_{64} * rb_{64}$



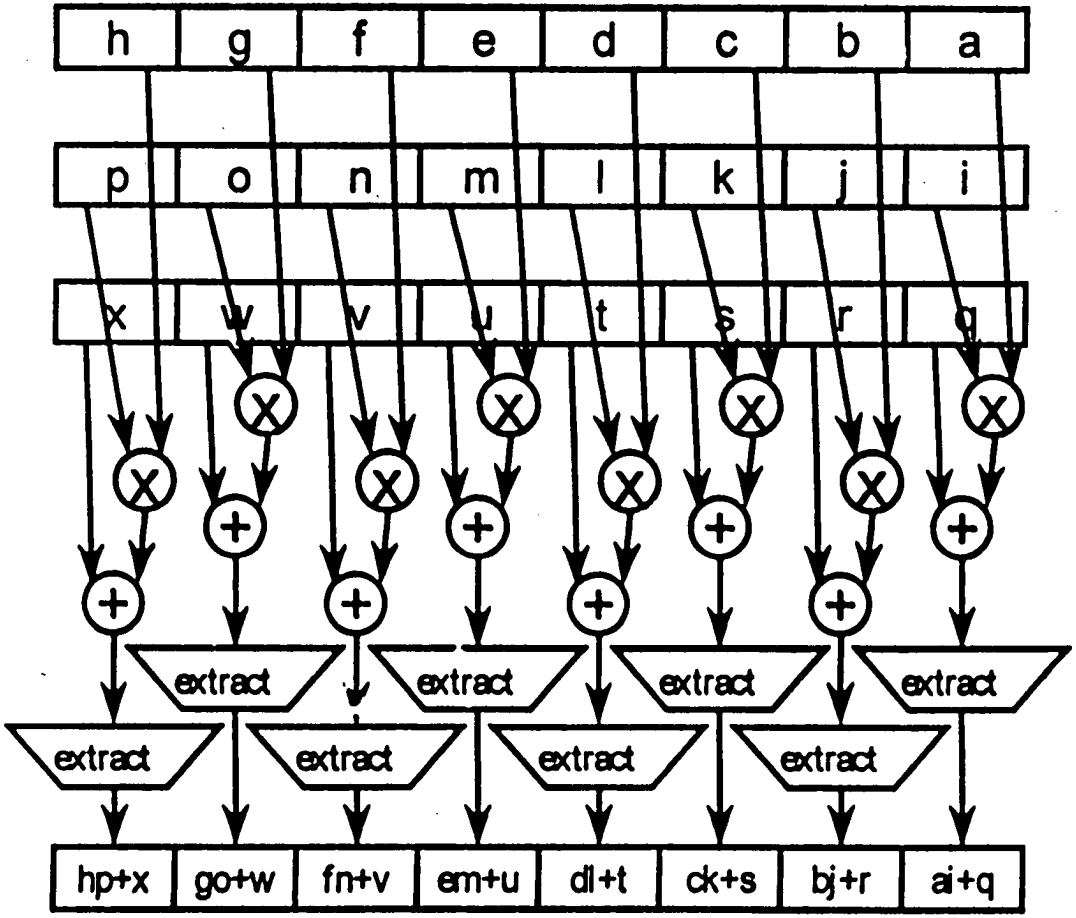
Ensemble multiply extract

- $rd_{128} = rc_{128} * rb_{128}$



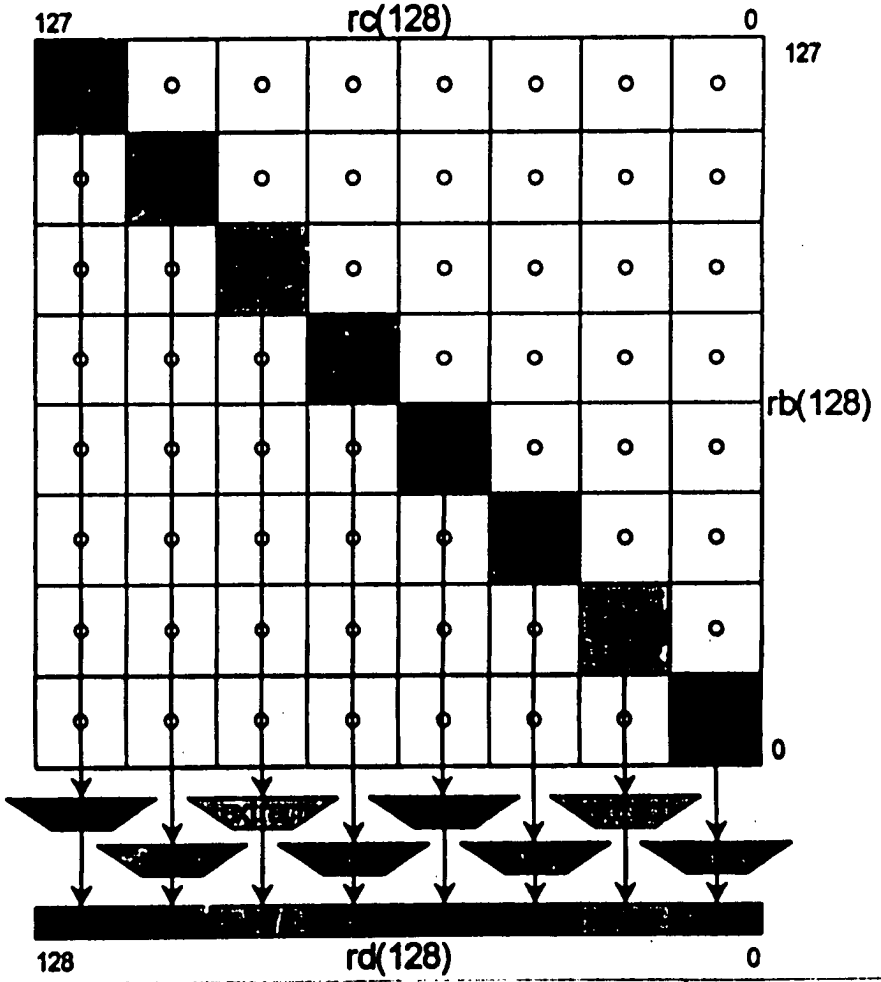
Ensemble multiply add extract

- $rd_{128} = rc_{128} * rb_{128} + rd_{128}$



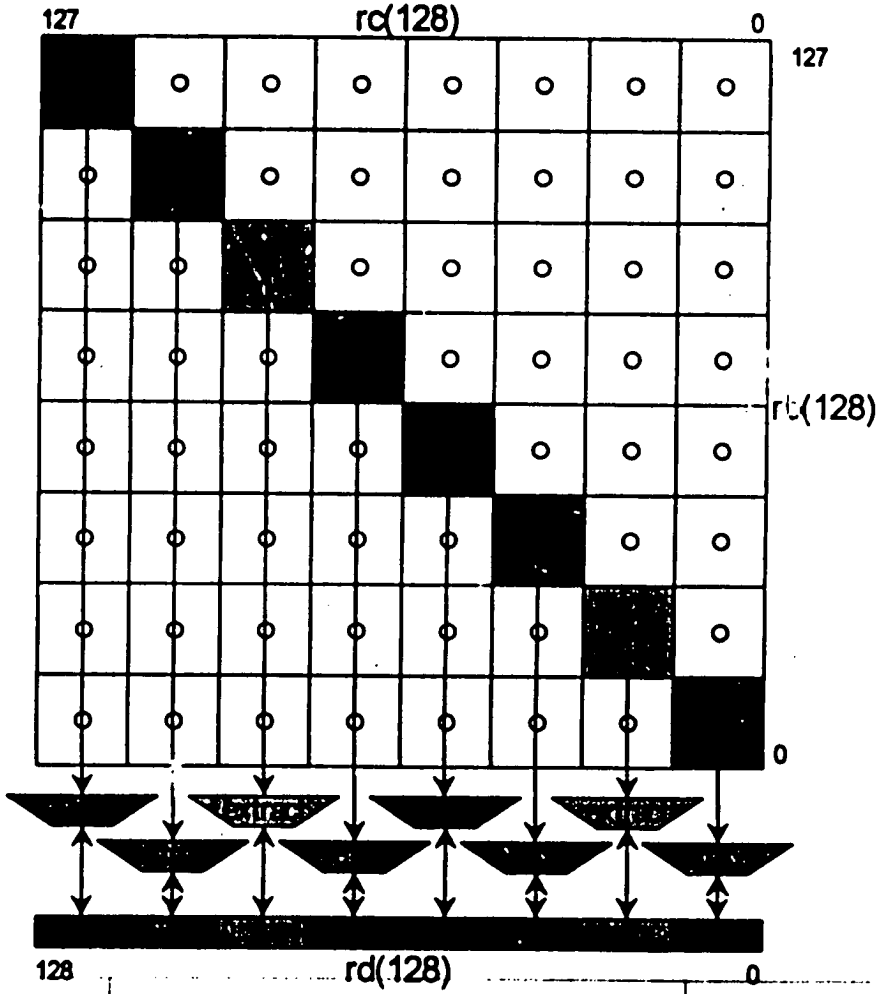
Ensemble multiply extract

- $rd_{128} = rc_{128} * rb_{128}$



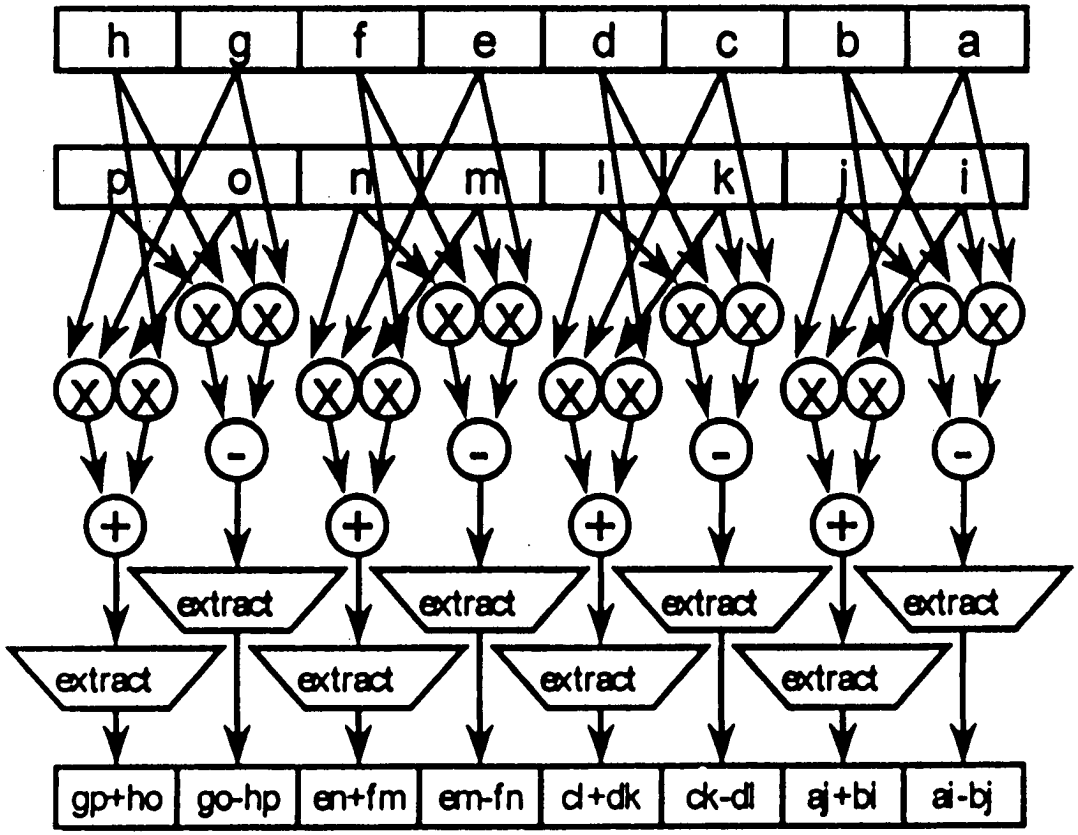
Ensemble multiply add extract

- $rd_{128} = rc_{128} * rb_{128} + rd_{128}$



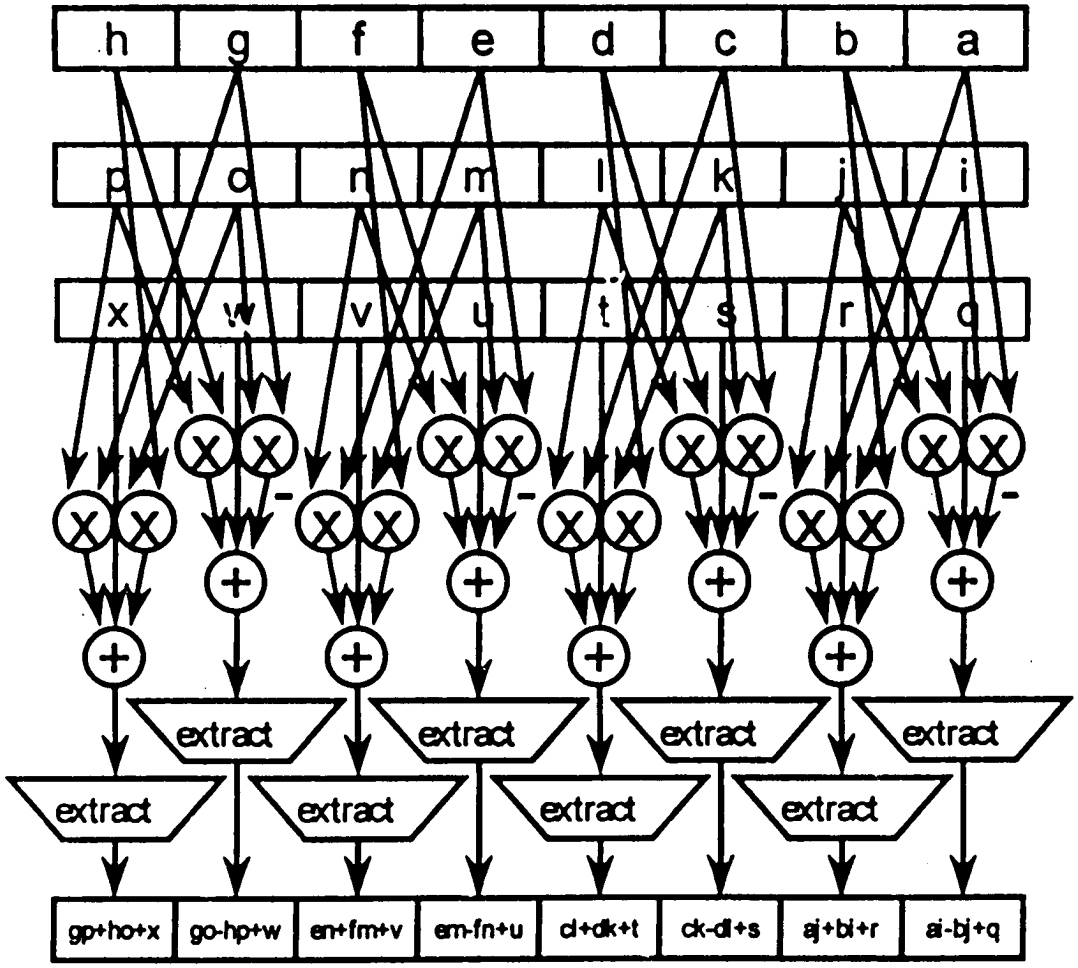
Ensemble multiply extract complex

- $rd_{128} = rc_{128} * rb_{128}$



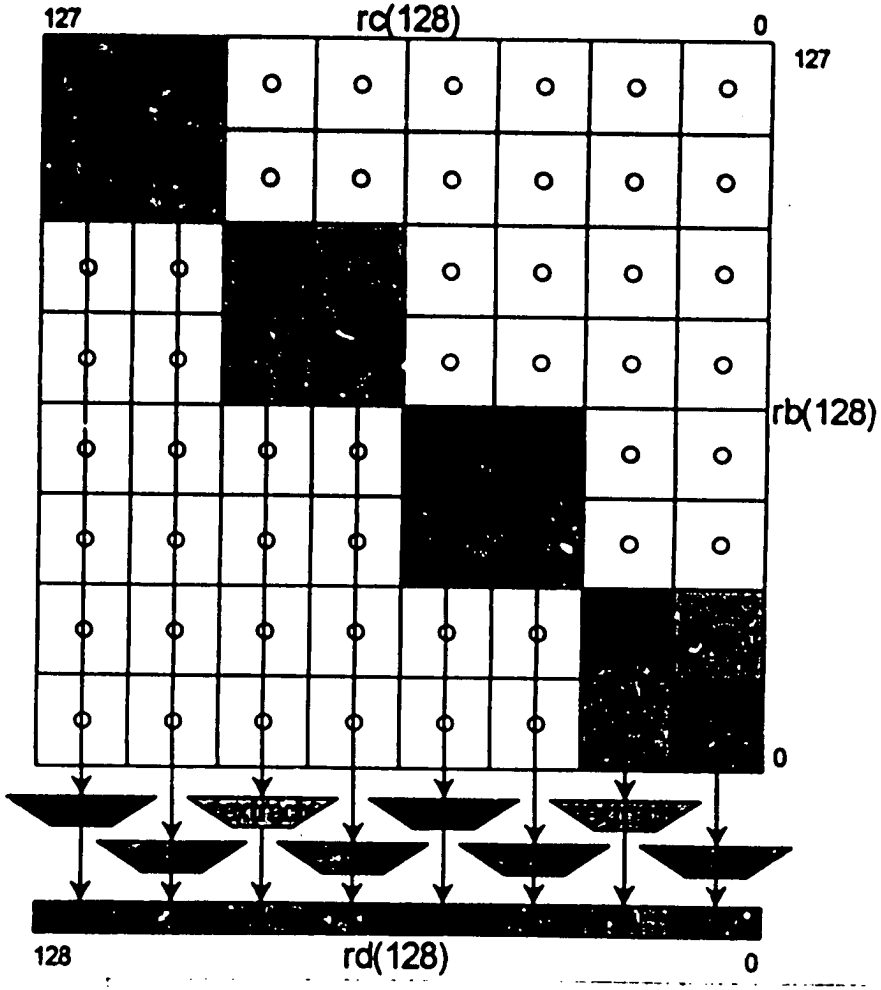
Ensemble multiply add extract complex

- $rd_{128} = rc_{128} * rb_{128} + rd_{128}$



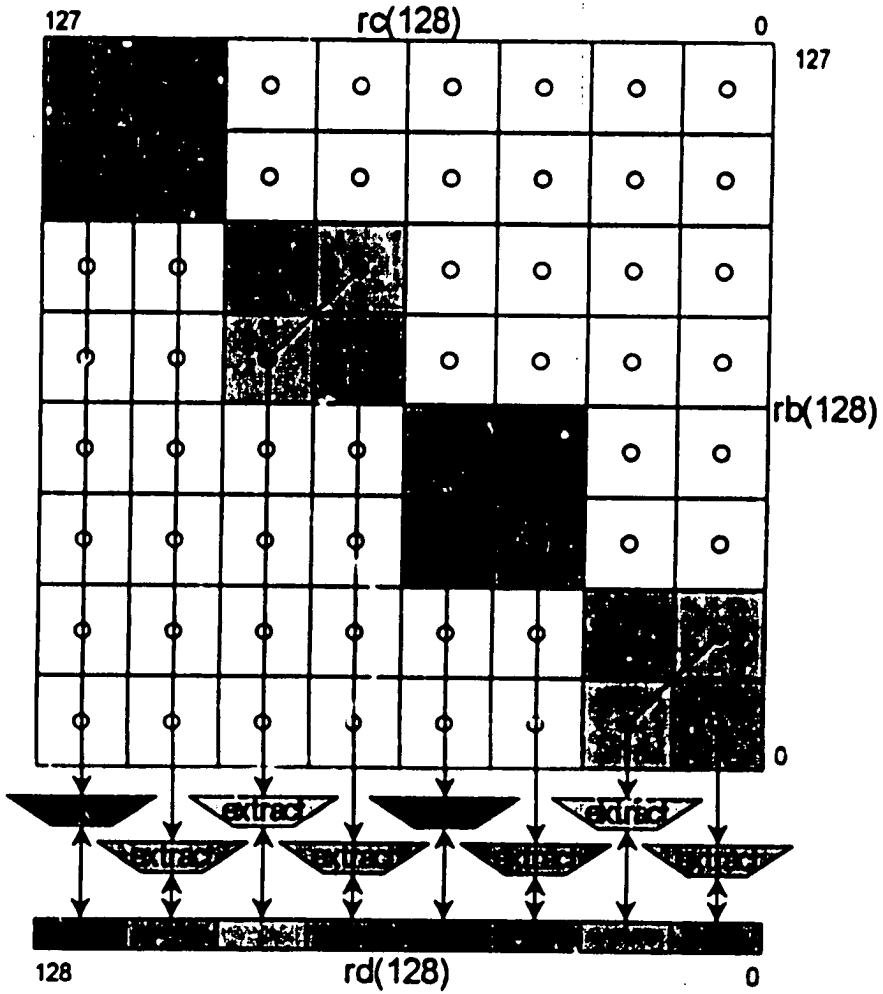
Ensemble multiply extract complex

- $rd_{128} = rc_{128} * rb_{128}$



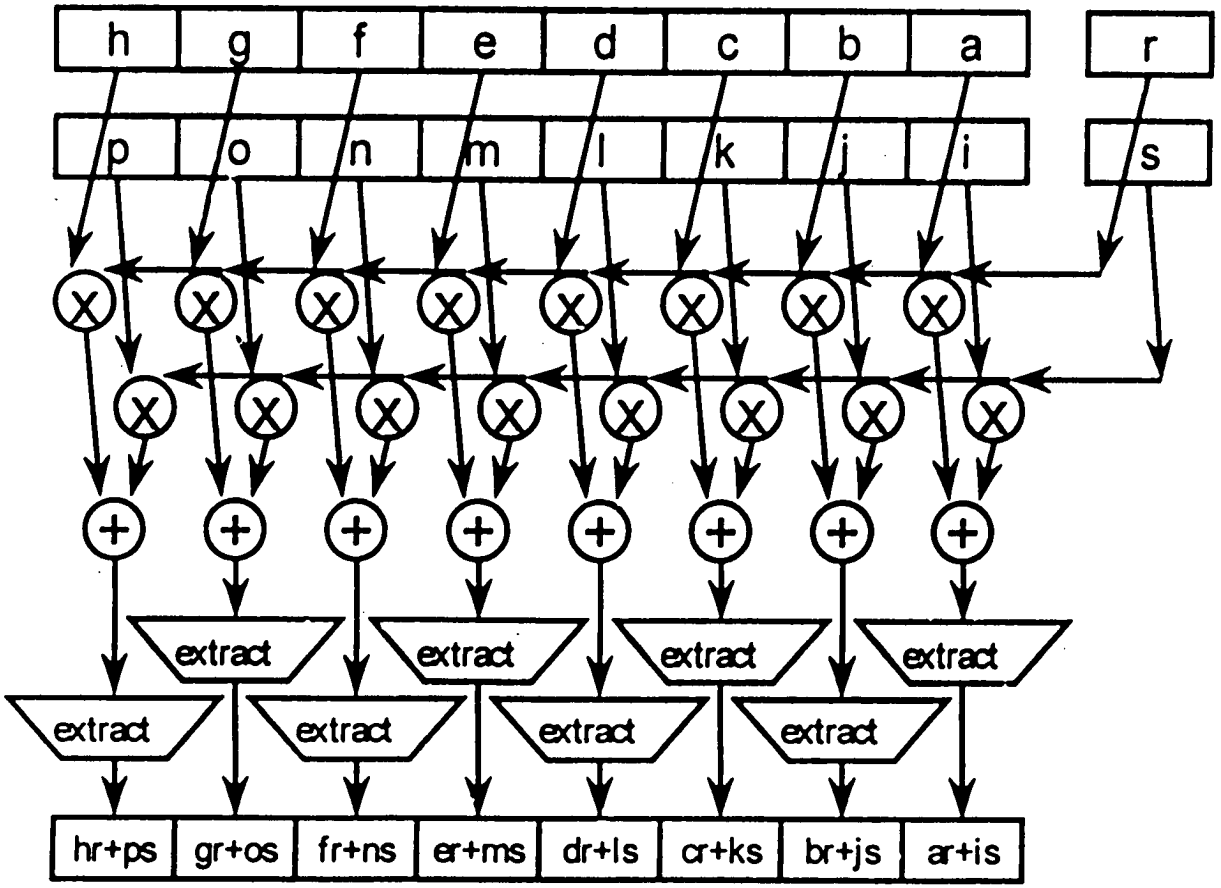
Ensemble multiply add extract complex

- $rd_{128} = rc_{128} * rb_{128} + rd_{128}$



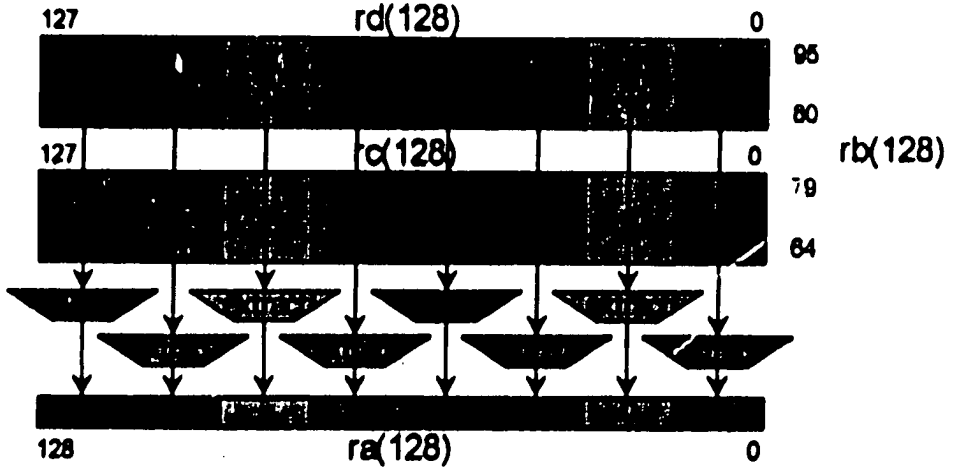
Ensemble scale add extract

- $ra_{128} = rd_{128} * rb_{size} + rc_{128} * rb_{size}$



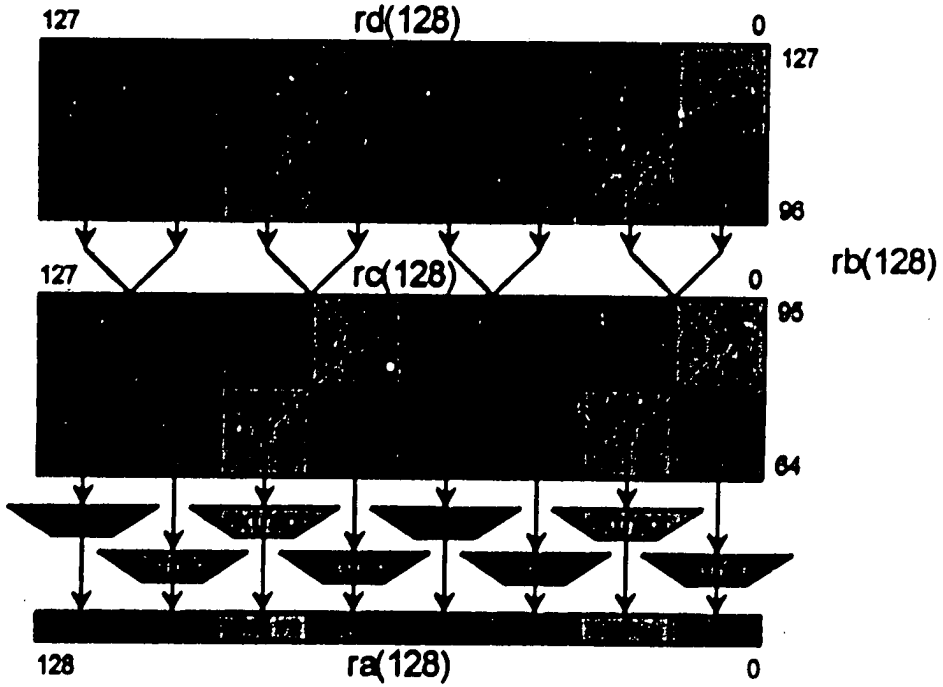
Ensemble scale add extract

- $ra_{128} = rd_{128} * rb_{size} + rc_{128} * rb_{size}$



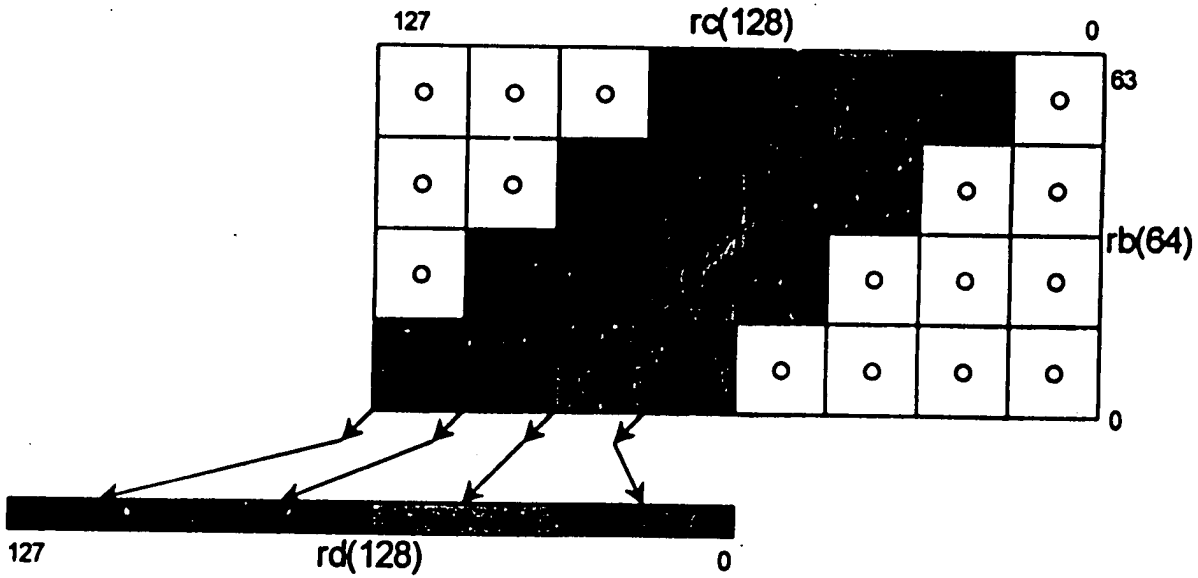
Ensemble scale add extract complex

- $ra_{128} = rd_{128} * rb_{size*2} + rc_{128} * rb_{size*2}$



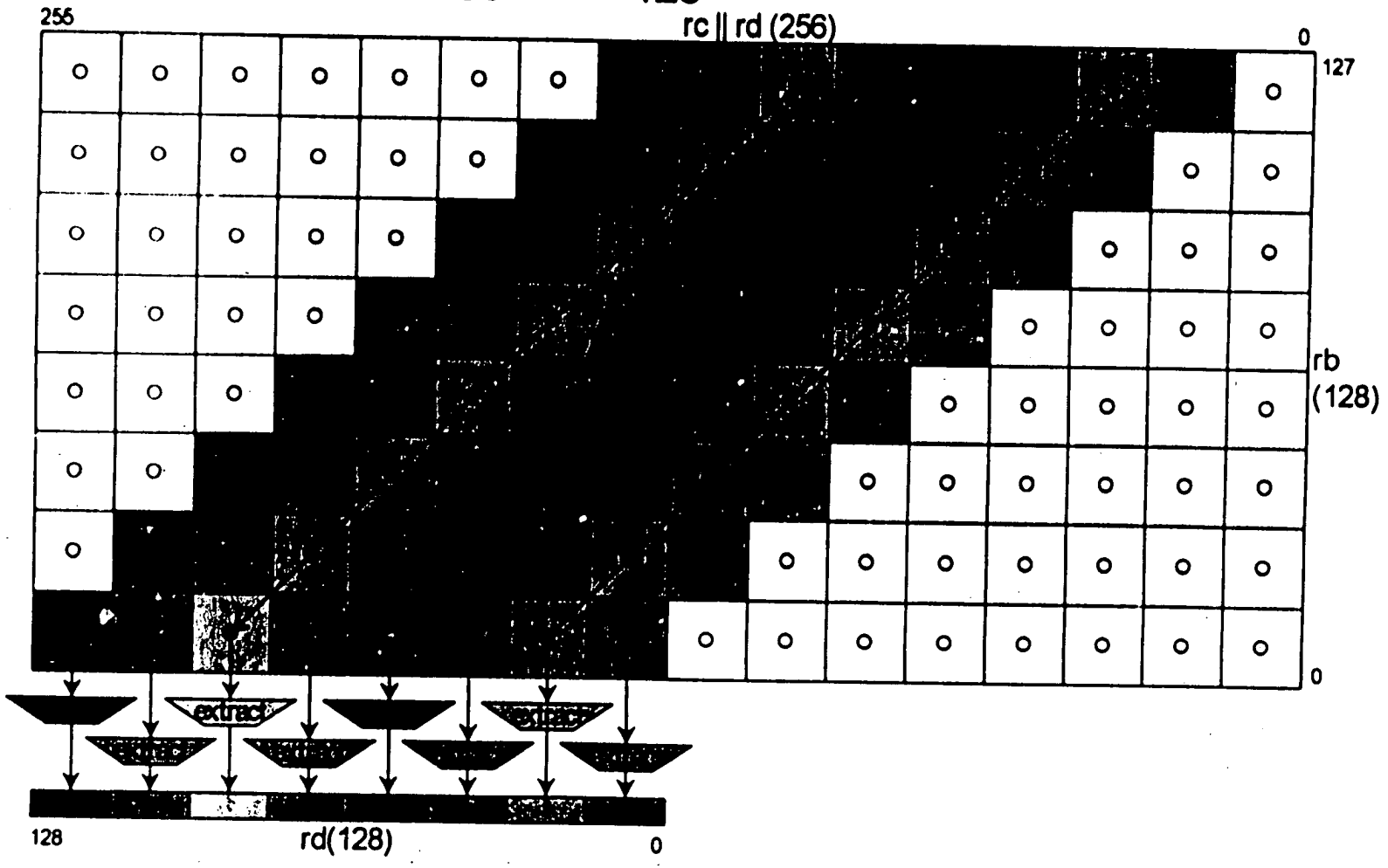
Ensemble convolve

- $rd_{128} = rc_{128} * rb_{64}$



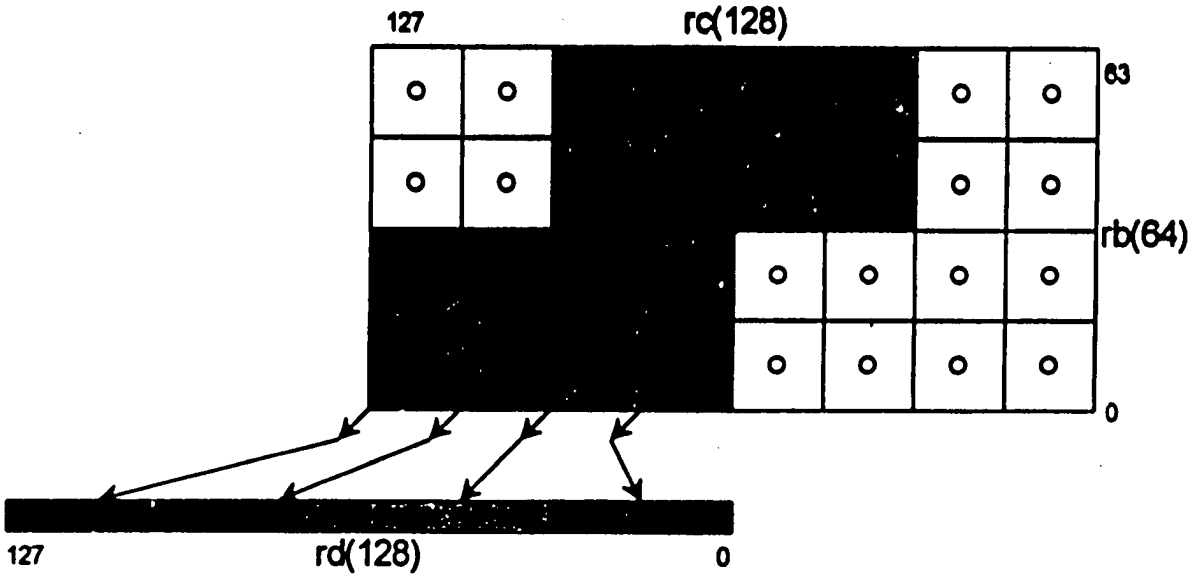
Ensemble convolve extract

- $rd_{128} = (rd||rc)_{256} * rb_{128}$



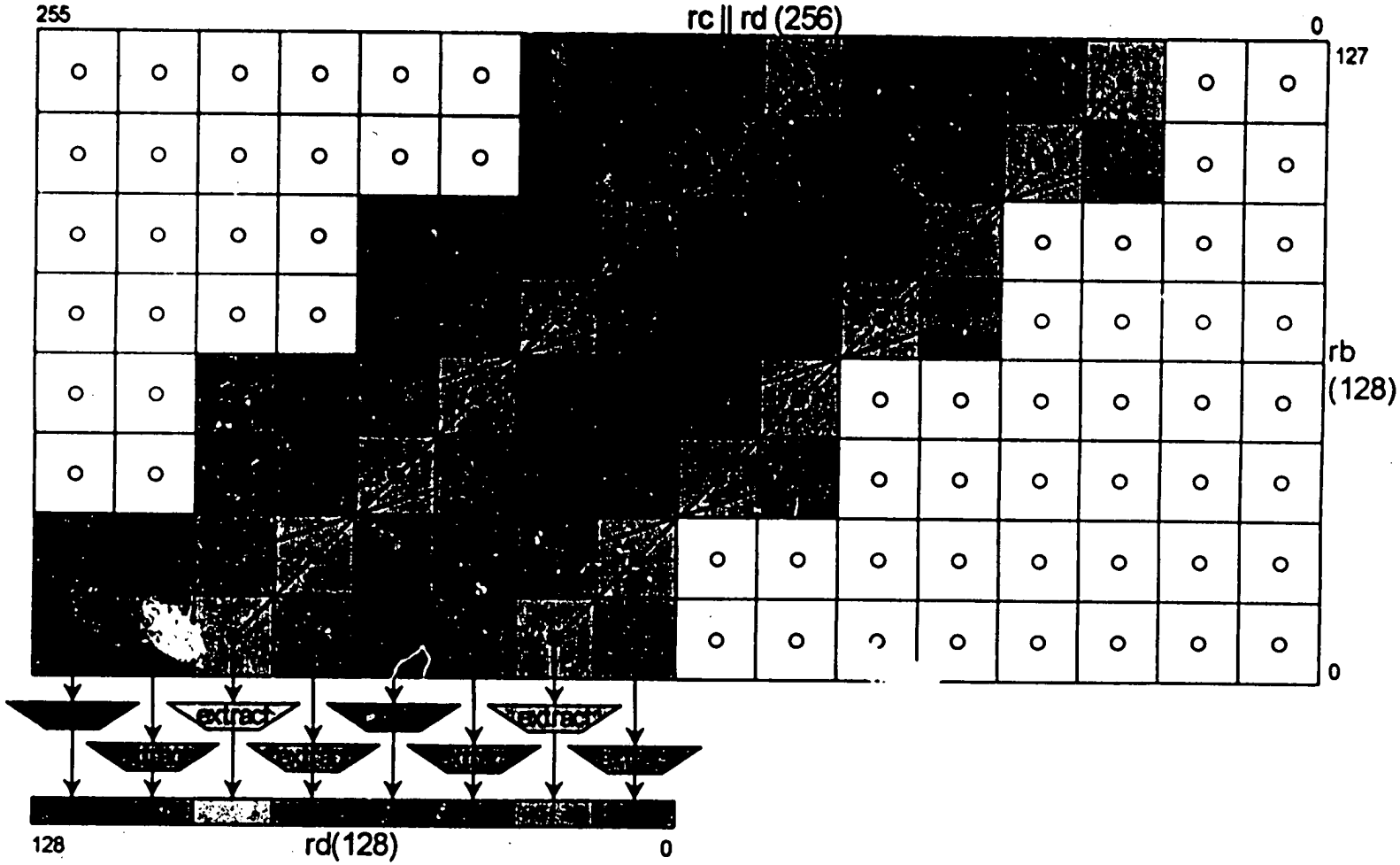
Ensemble convolve complex

- $rd_{128} = rc_{128} * rb_{64}$



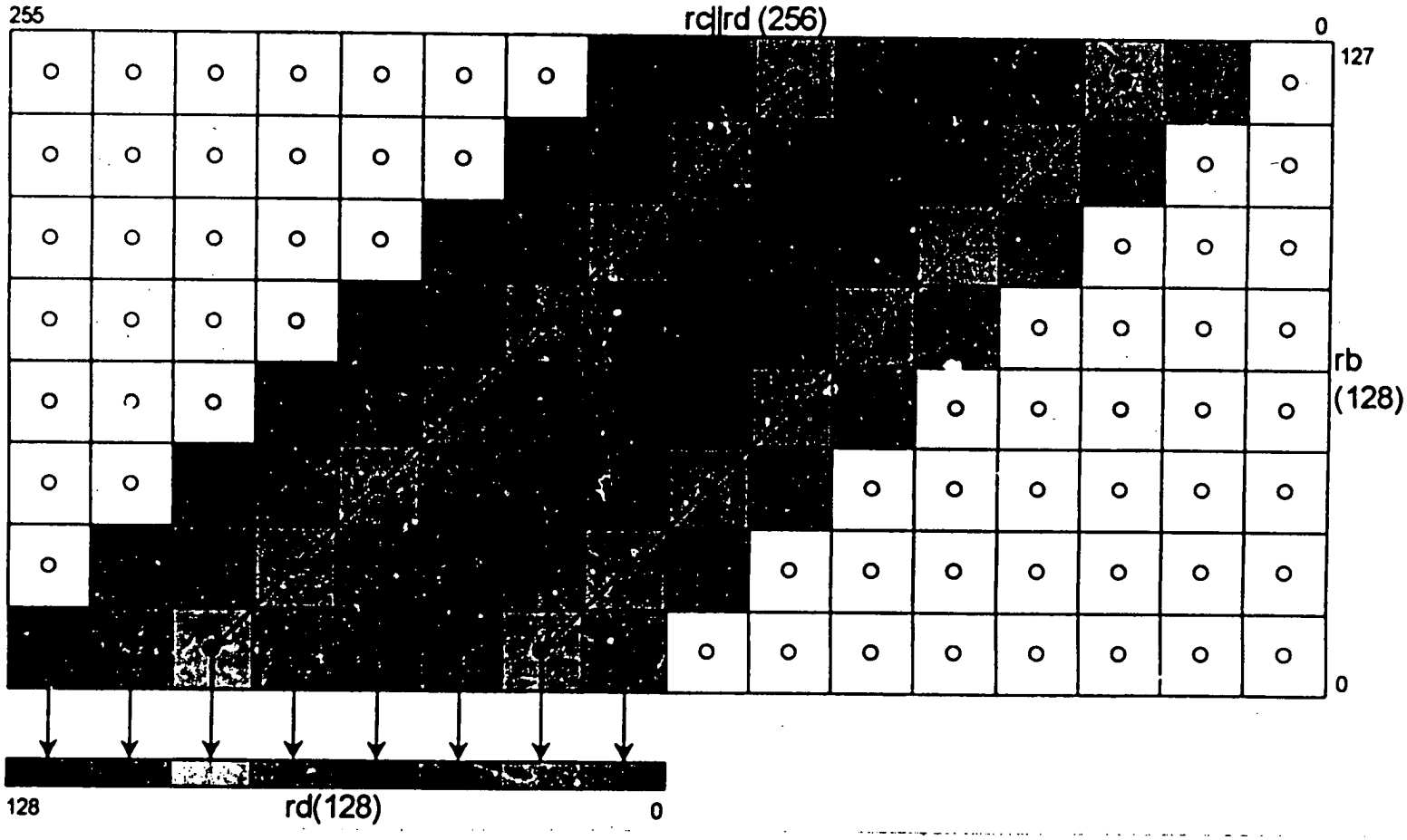
Ensemble convolve extract complex

- $rd_{128} = (rc||rd)_{256} * rb_{128}$



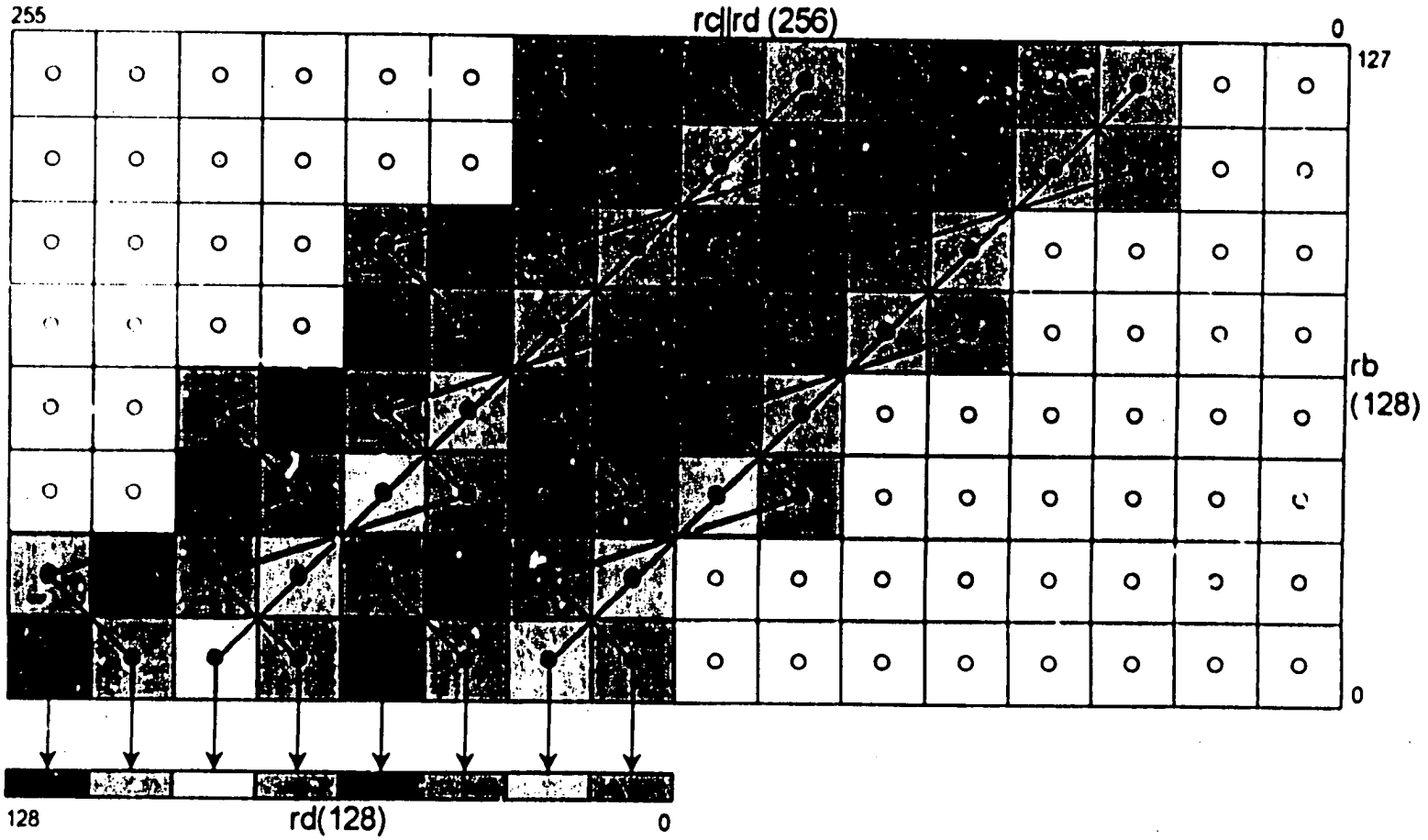
Ensemble convolve floating-point

- $rd_{128} = (rc||rd)_{256} * rb_{128}$



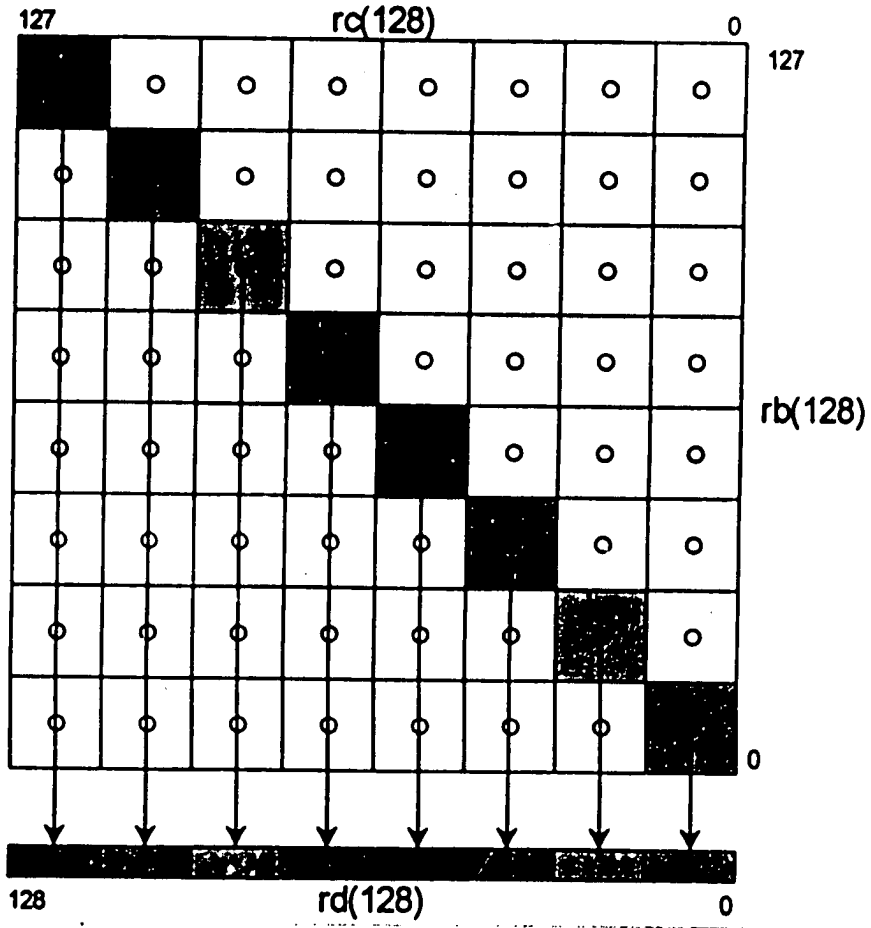
Ensemble convolve complex floating-point

- $rd_{128} = (rc||rd)_{256} * rb_{128}$



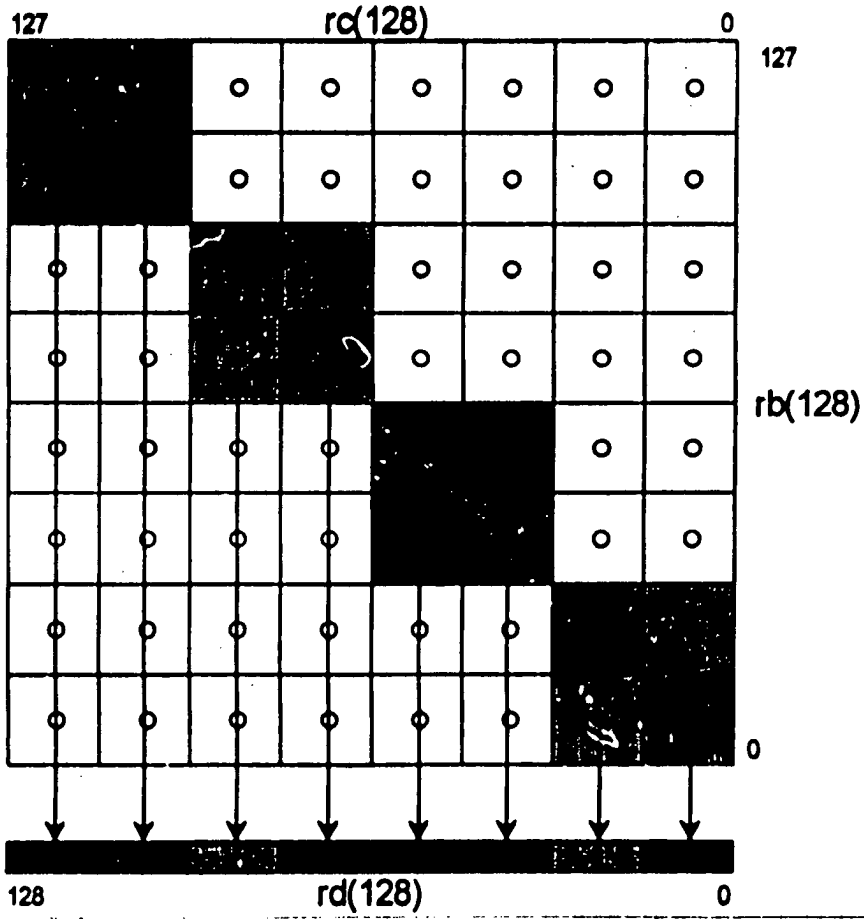
Ensemble multiply floating-point

- $rd_{128} = rc_{128} * rb_{128}$



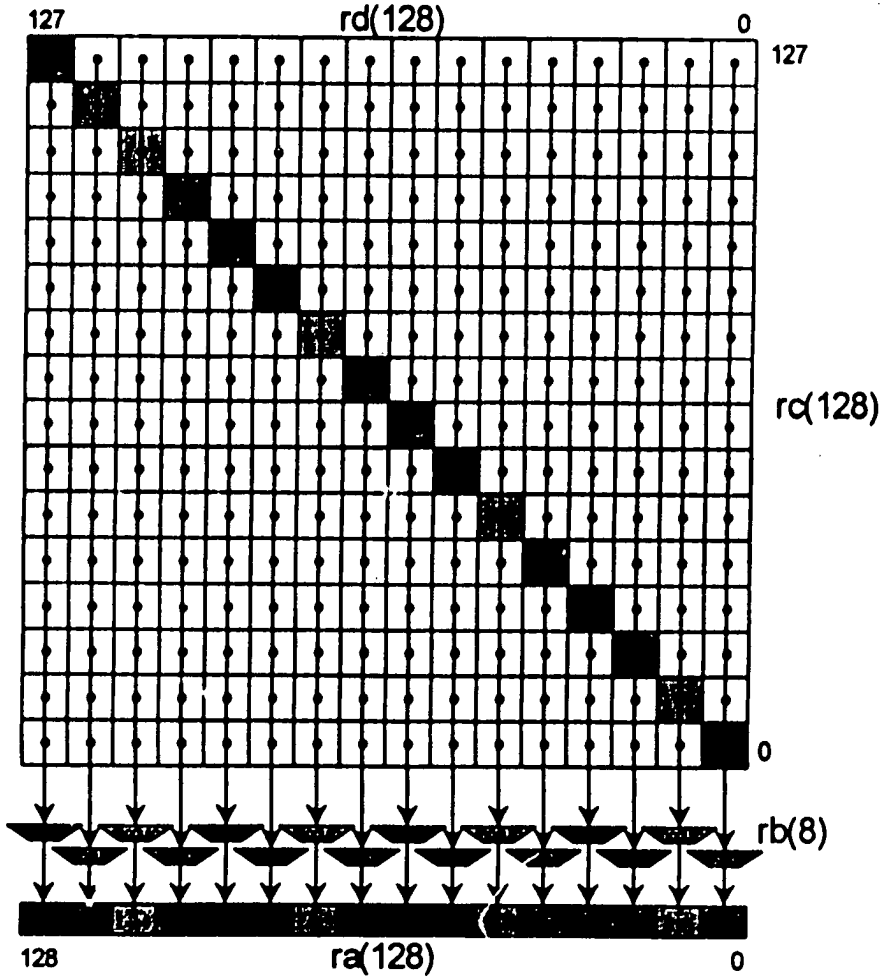
Ensemble multiply floating-point complex

- $rd_{128} = rc_{128} * rb_{128}$



Ensemble multiply Galois

- $ra_{128} = rd_{128} * rc_{128} \bmod rb_8$



Wide Instructions

- Wide Multiply Matrix E
- Wide Switch X
- Wide Table L

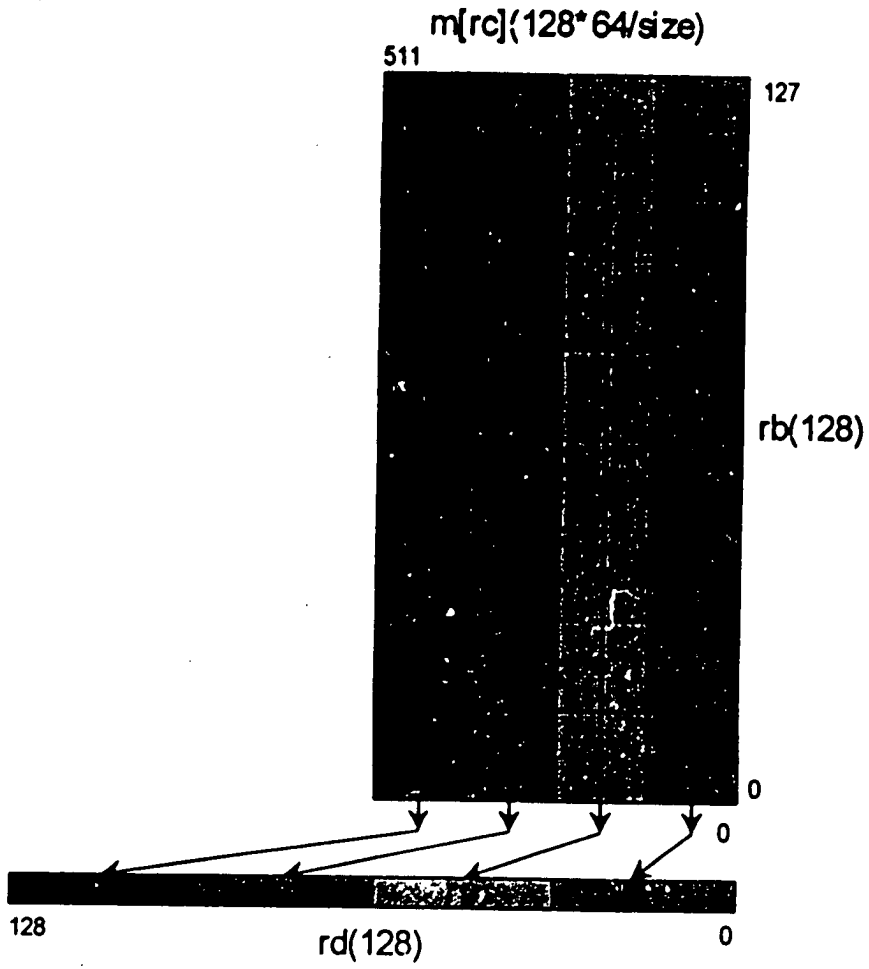
Wide size and shape

- operations up to 128x128
- full size not always required
- optional bits set in address
  - sets operand size
  - sets operand width
- operand aligned to specified size

- smaller size *may* use fewer cycles
  - to load operand cache
  - to perform operation

Wide multiply matrix

- $rd_{128} = m[rc]_{(128*64/size)} * rb_{128}$

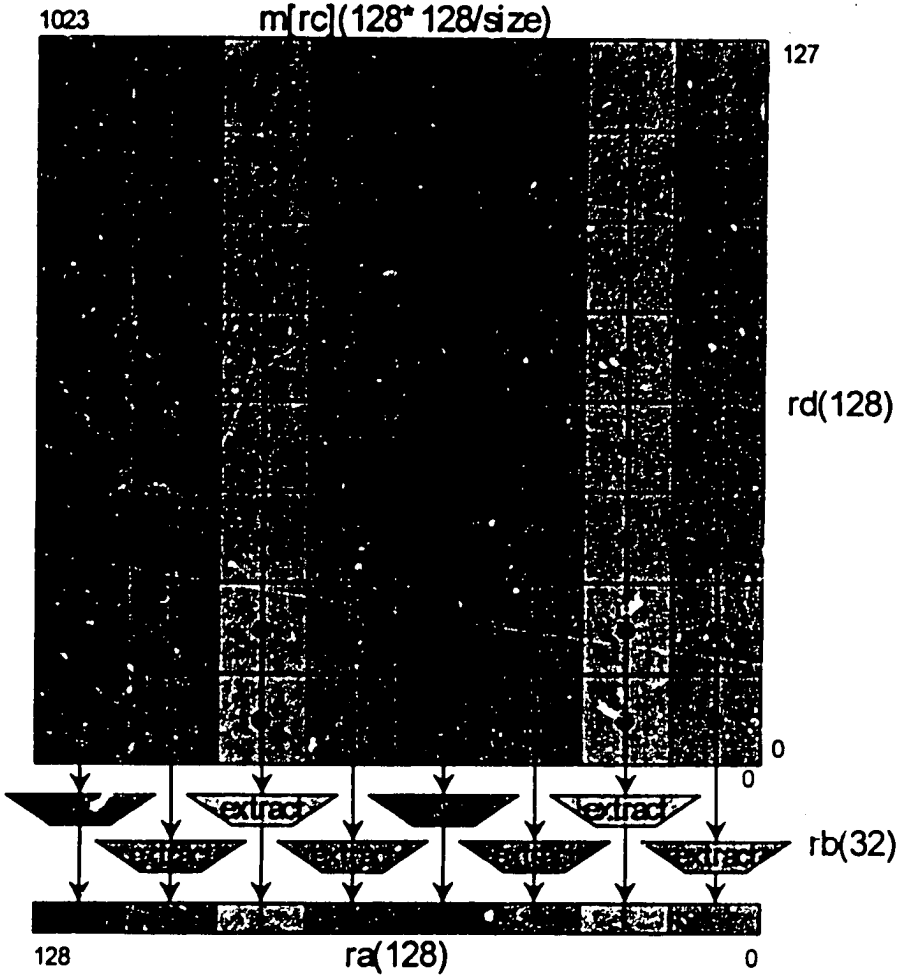


Wide multiply matrix complex

- $rd_{128} = m[rc]_{(64*64/size)} * rb_{128}$

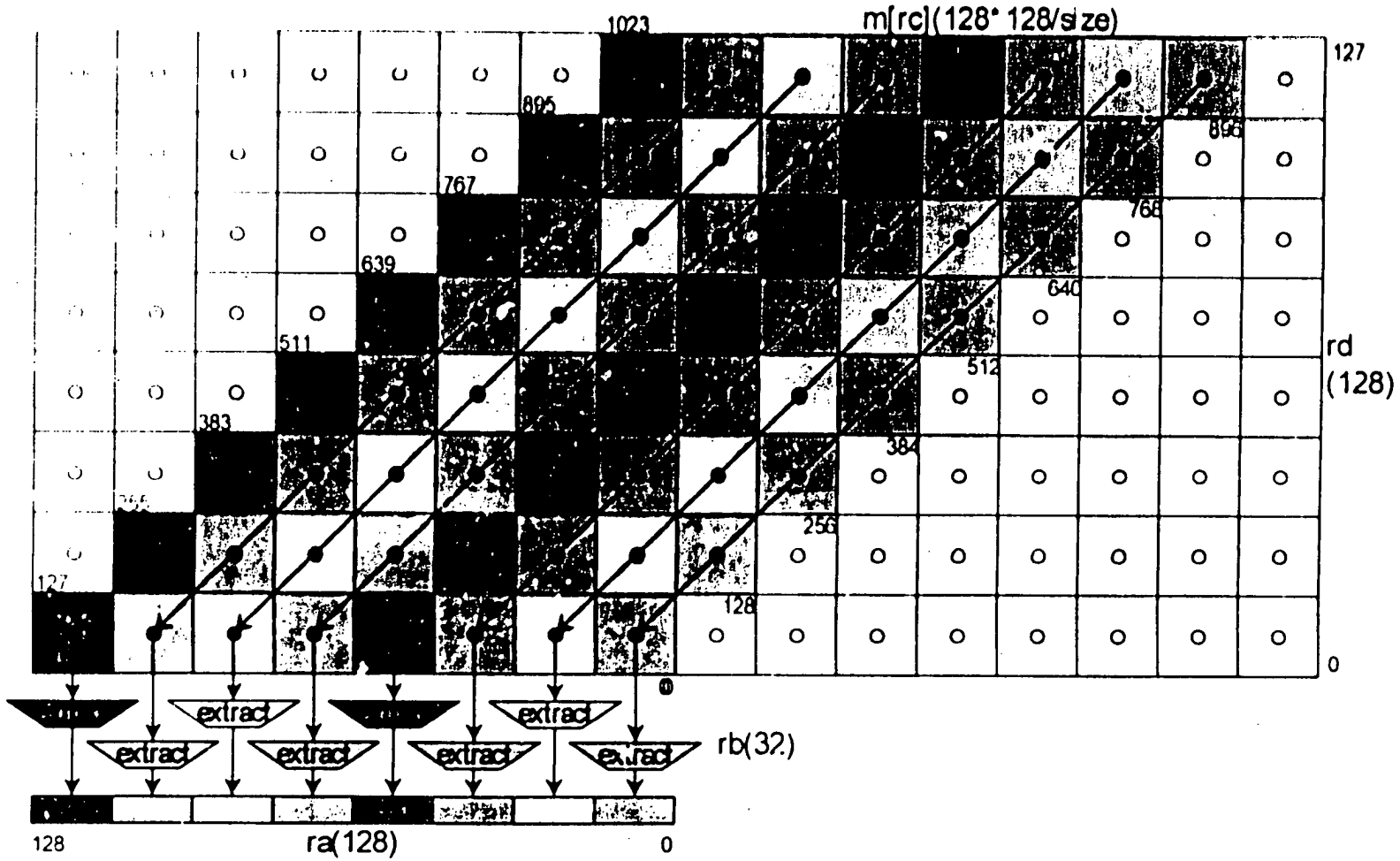
Wide multiply matrix extract

- $ra_{128} = m[rc]_{128*128/size} * rd_{128}$



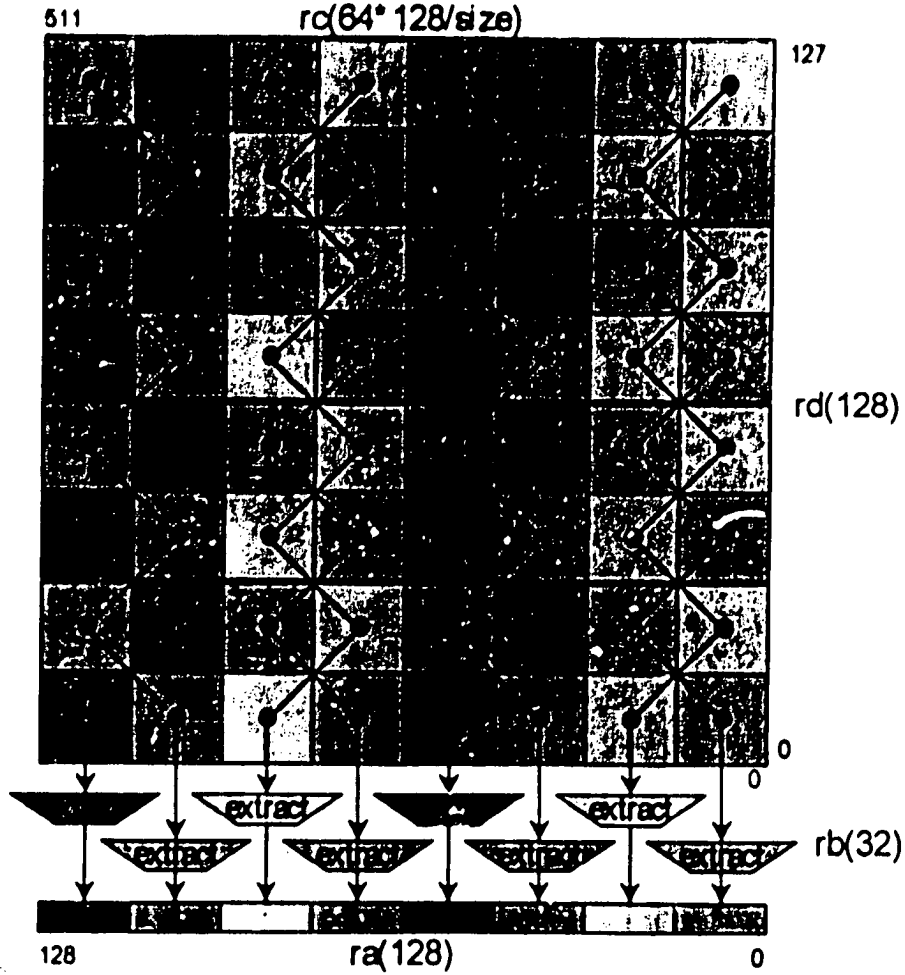
Wide multiply matrix extract

- $ra_{128} = m[rc]_{128*128/size} * rd_{128}$



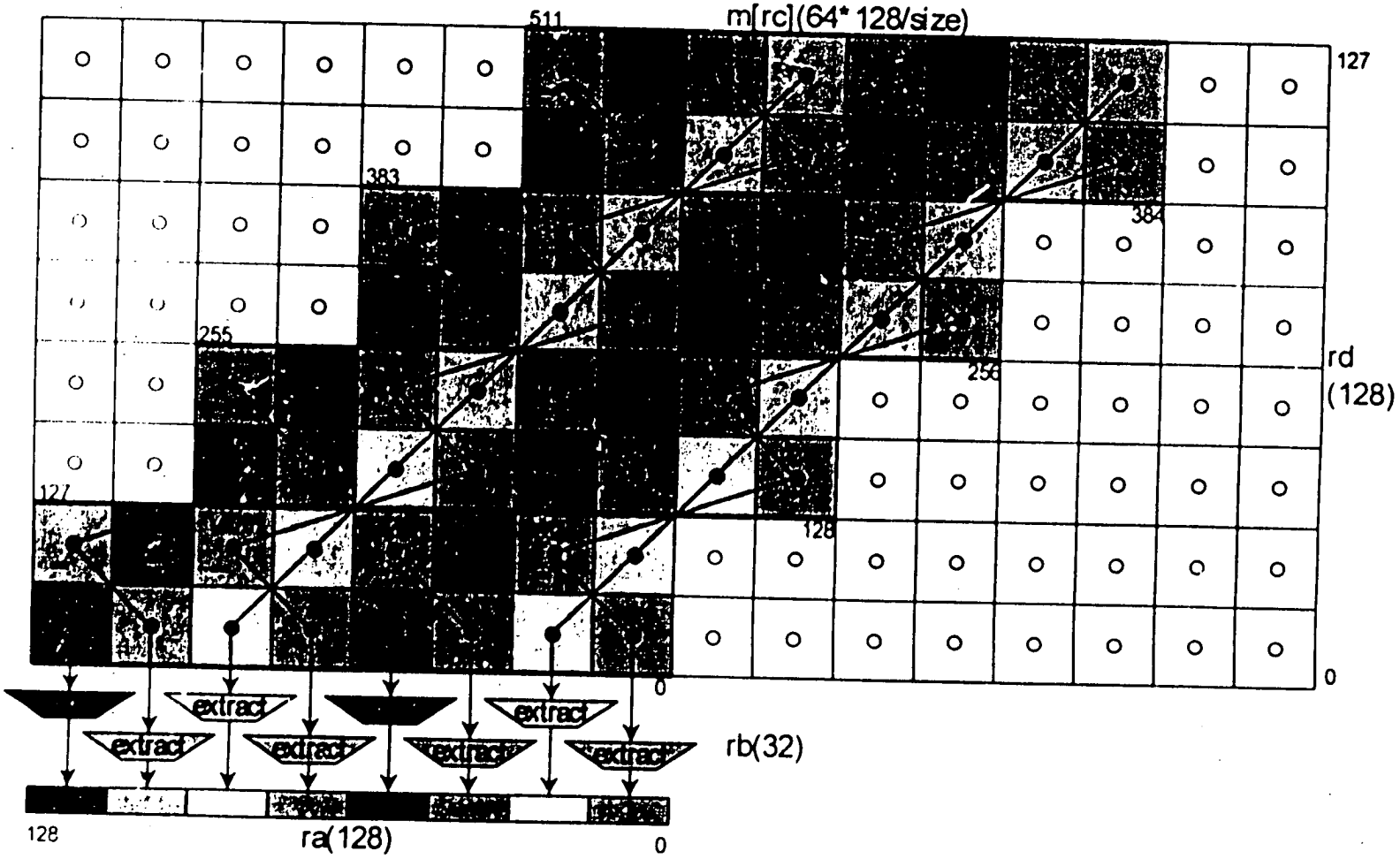
Wide multiply matrix extract complex

- $ra_{128} = m[rc]_{64*128/size} * rd_{128}$



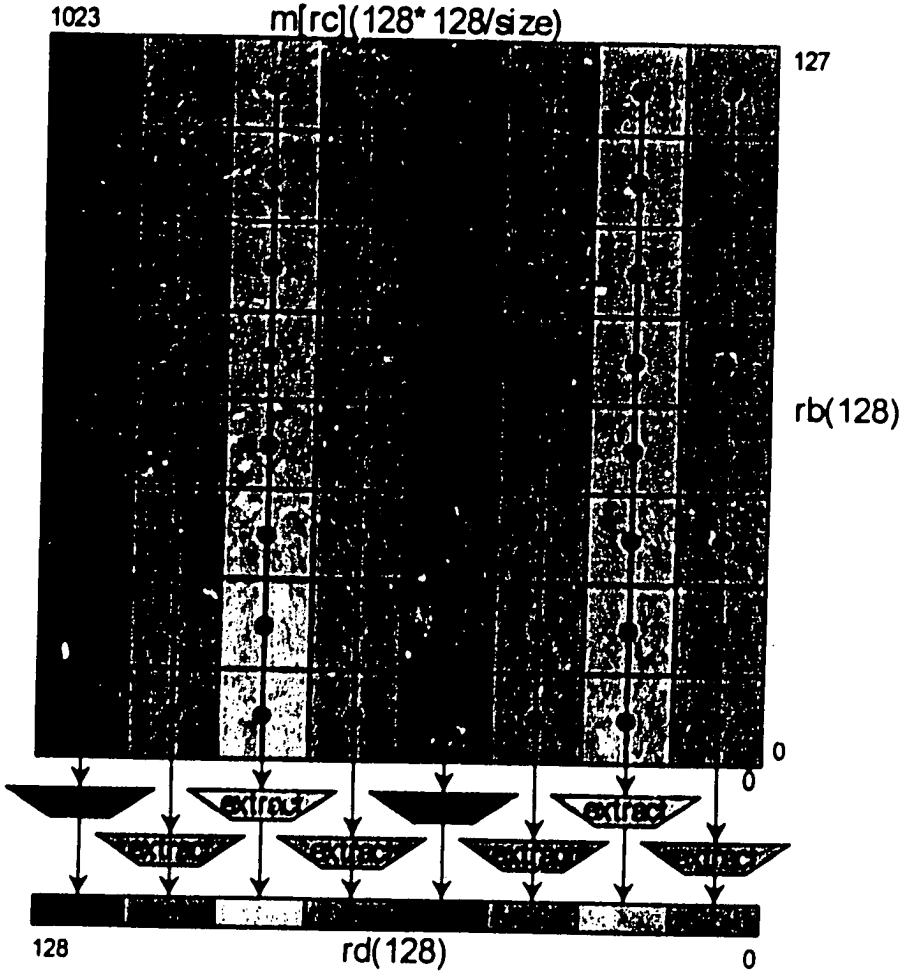
Wide multiply extract complex

- $ra_{128} = m[rc]_{64*128/size} * rd_{128}$



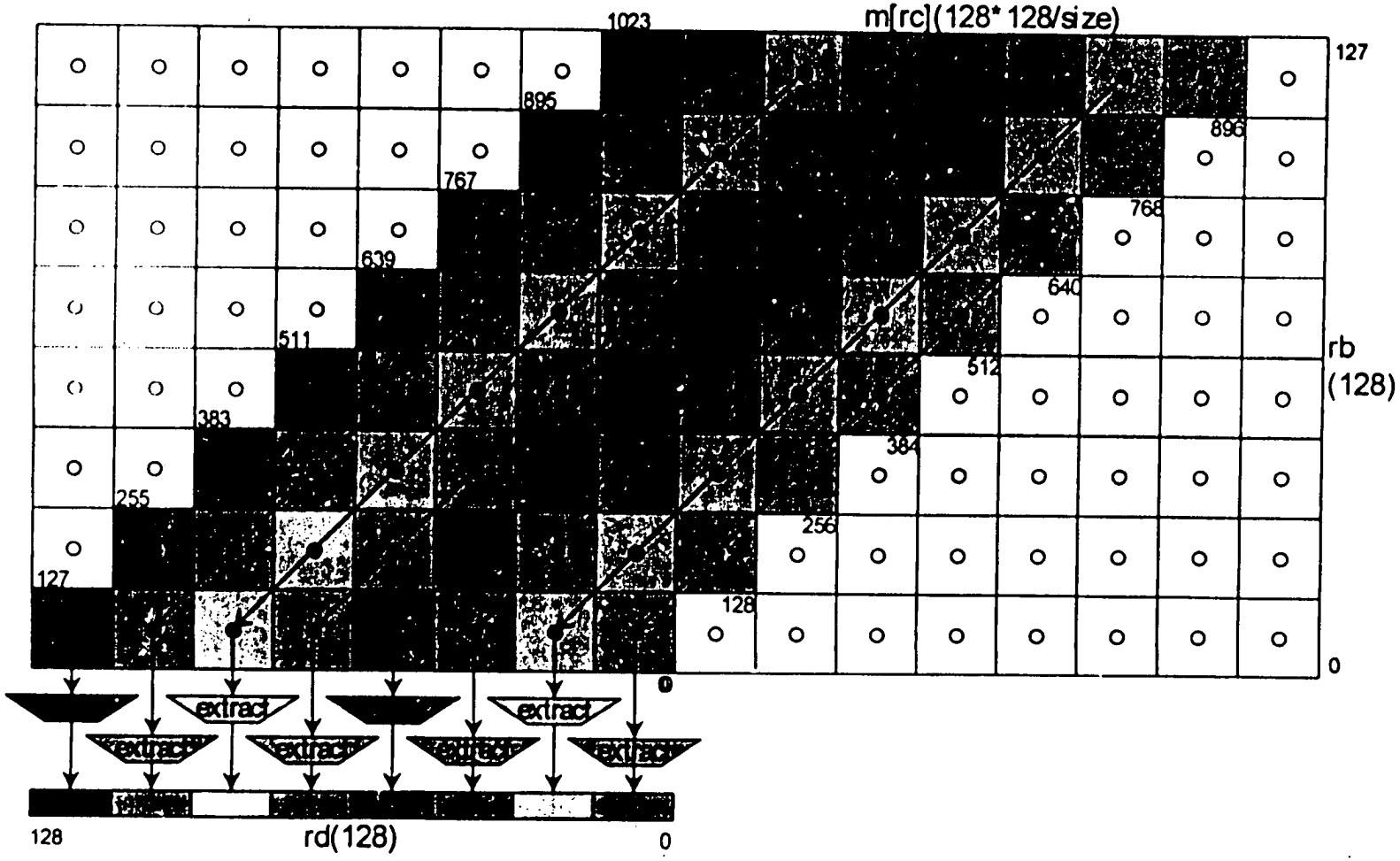
Wide multiply matrix extract immediate

- $rd_{128} = m[rc]_{128*128/size} * rb_{128}$



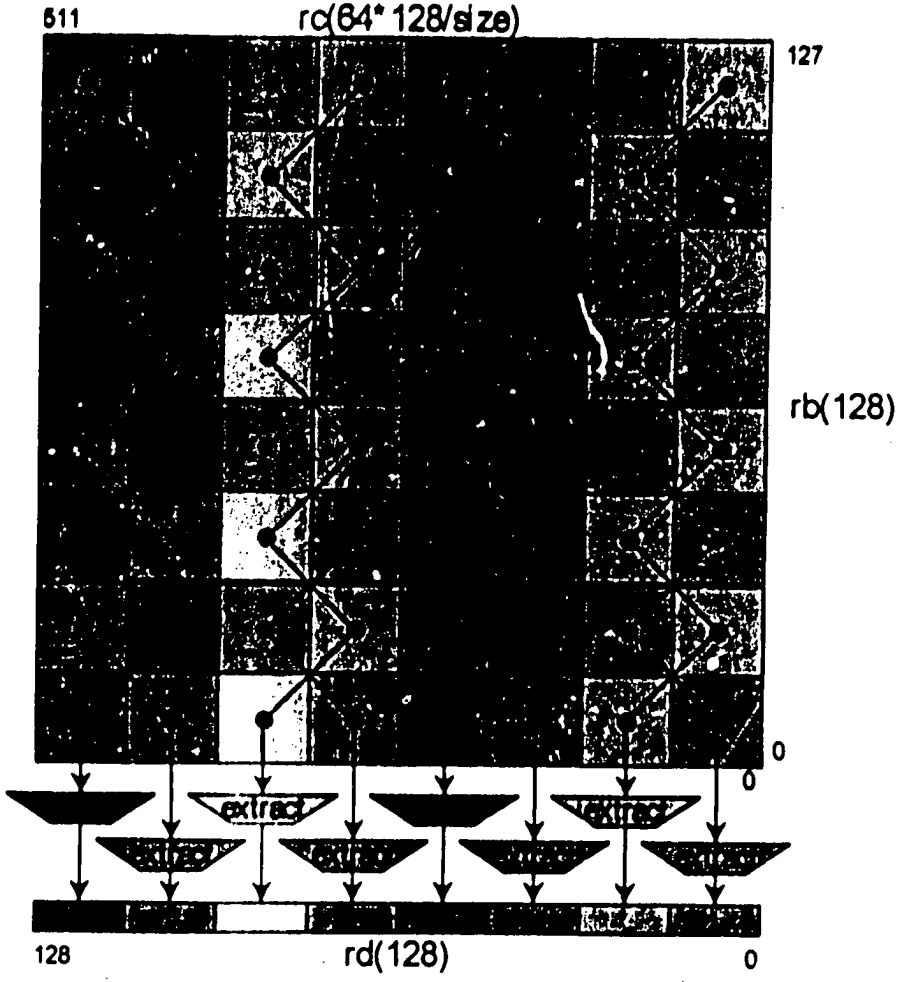
Wide multiply matrix extract immediate

- $rd_{128} = m[rc]_{128*128/size} * rb_{128}$



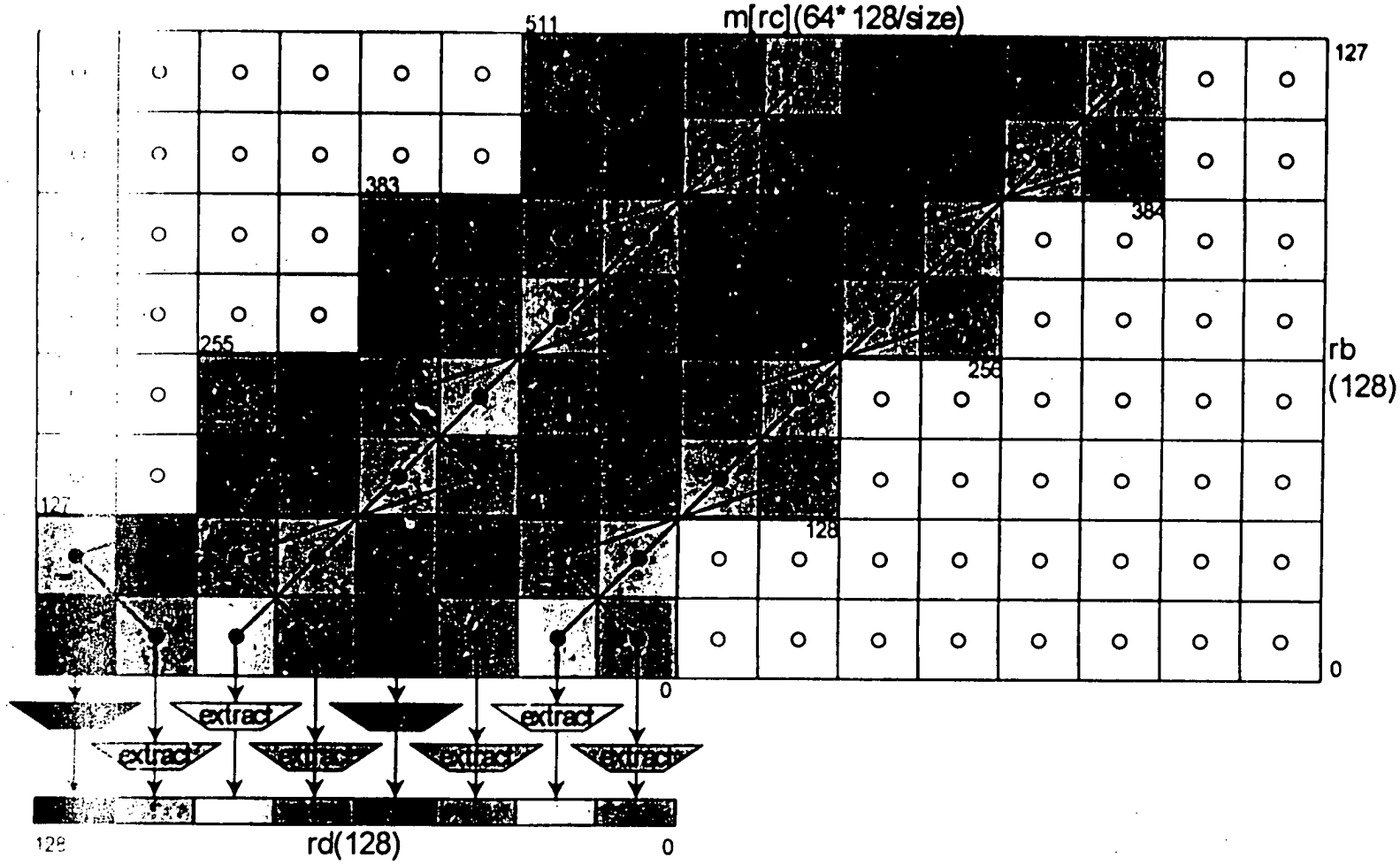
Wide multiply matrix extract immediate complex

- $rc_{128} = m[rc]_{64*128/size} * rb_{128}$



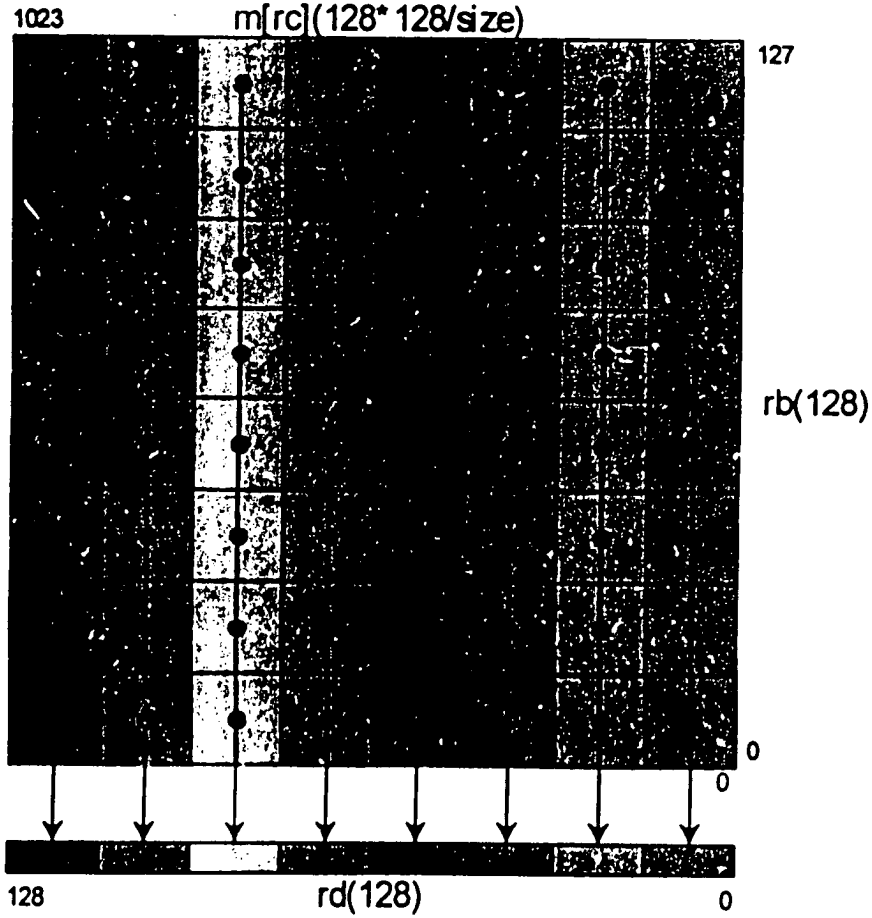
Wide multiply extract immediate complex

- $rd_{128} = m[rc]_{64*128/size} * rb_{128}$



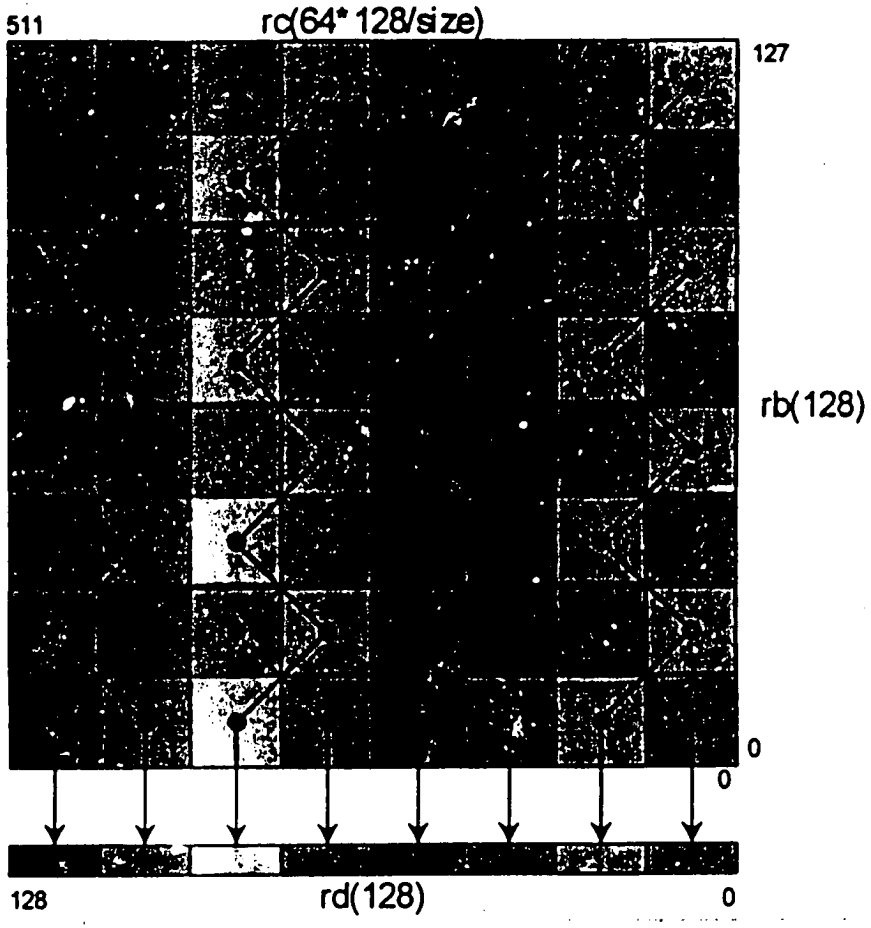
Wide multiply matrix floating-point

- $rd_{128} = m[rc]_{128*128/size} * rb_{128}$



Wide multiply matrix complex floating-point

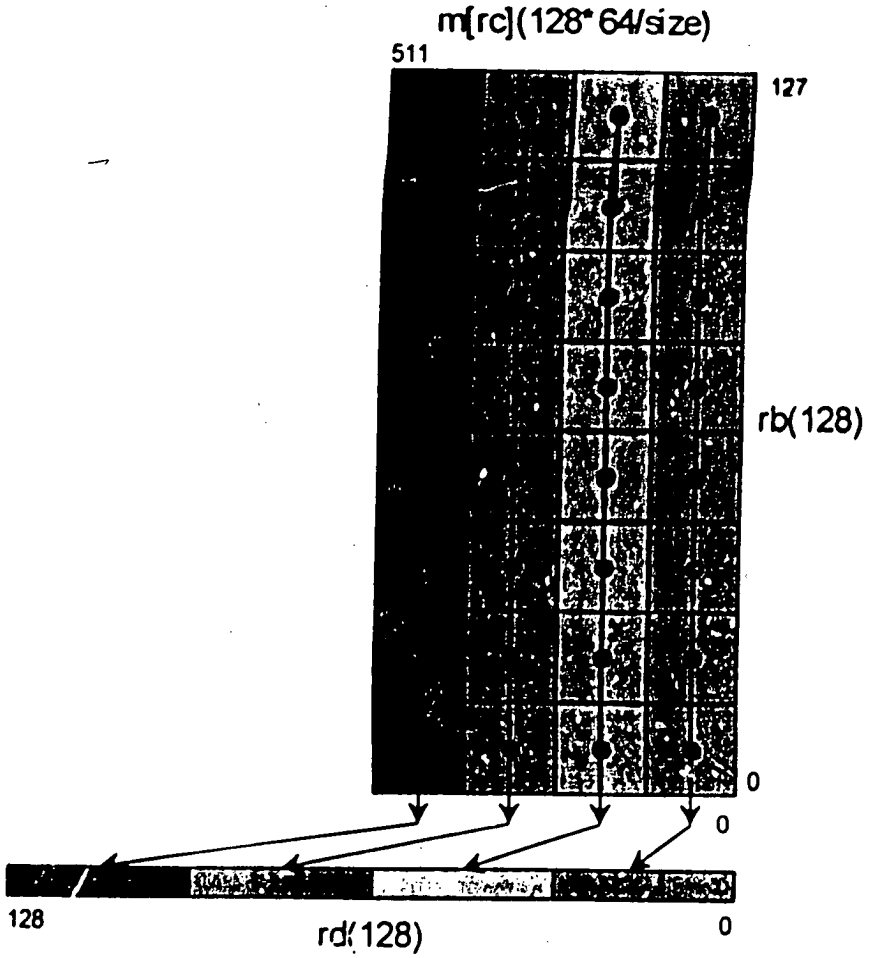
- $rd_{128} = m[rc]_{64*128/size} * rb_{128}$



MicroUnity

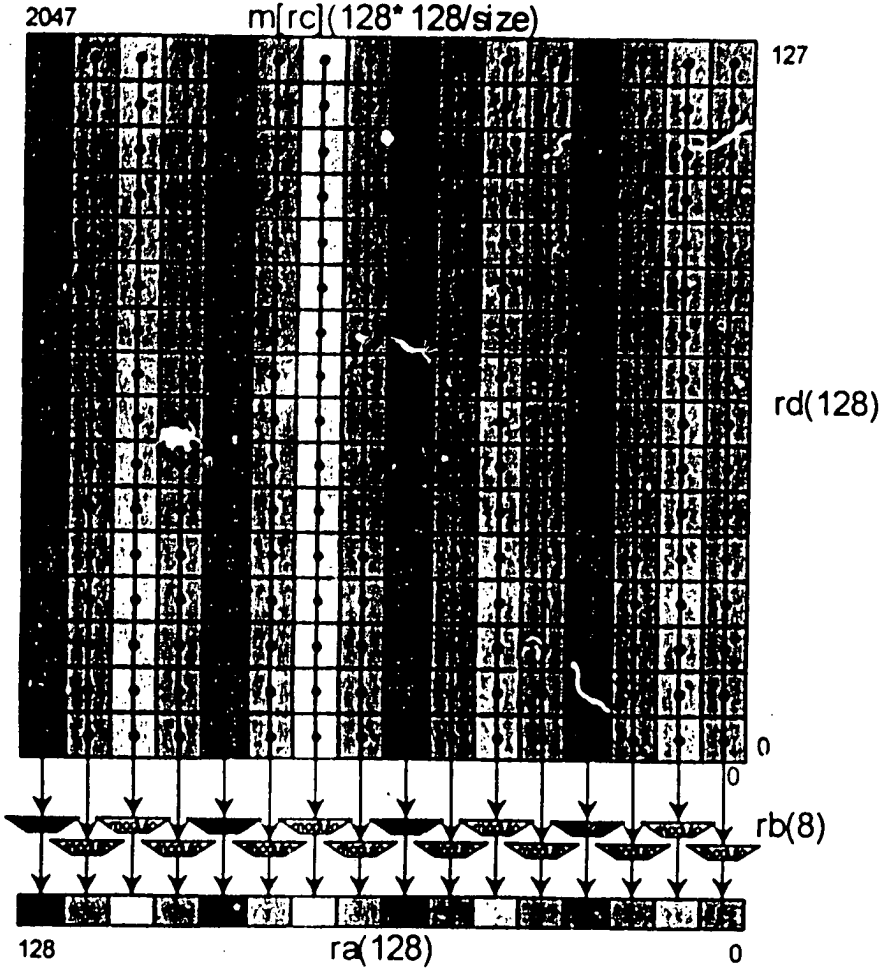
Wide multiply matrix polynomial

- $rd_{128} = m[rc]_{(128*64/size)} * rb_{128}$



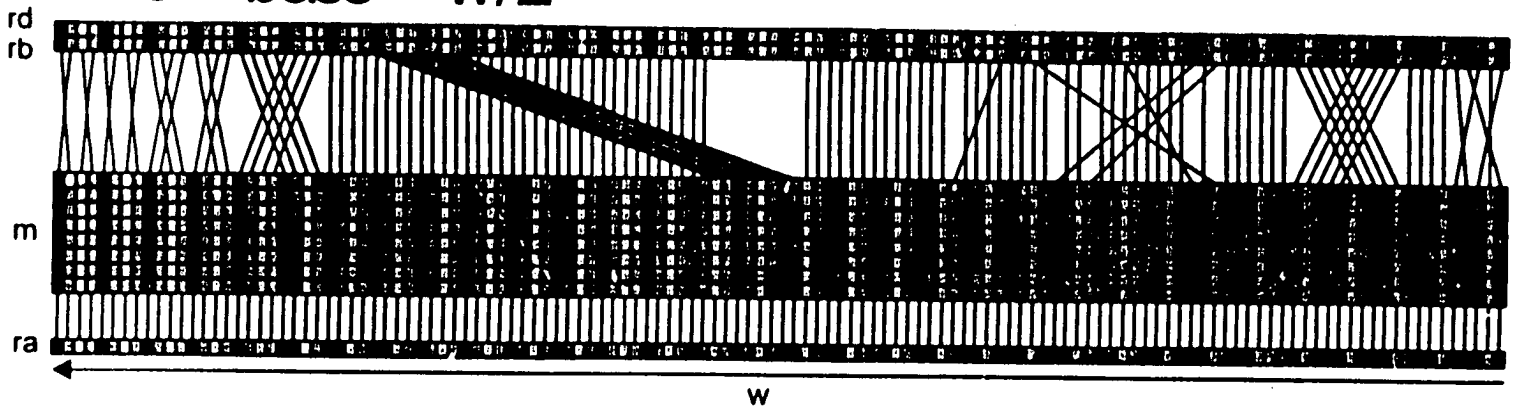
Wide multiply matrix Galois

- $ra_{128} = m[rc]_{128*128/size} * rd_{128} \bmod rb_8$



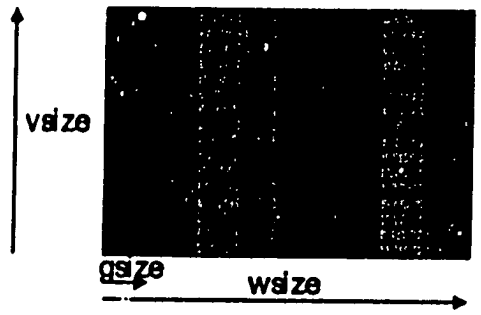
Wide switch

- $j(i) = m[rc]_{7w+i,6w+i,5w+i,4w+i,3w+i,2w+i,w+i,i}$
- $ra_i = (rd||rb)_j$, i=0..127
- rc specifies address and w
  - rc = base + w/2



Wide Table

- Table lookup
  - msize: total table size
  - wsize: table width
  - vsize: table depth
  - gsize: Group size (table granularity)



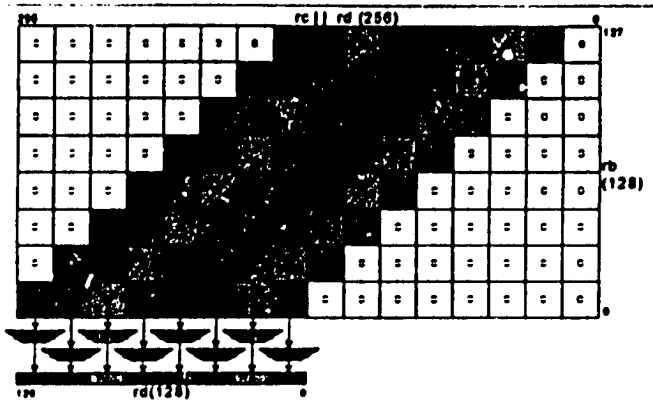
- $j(i) = b_{|vsize-1+i..i*wsize+i|wsize-1..0}$
- $rd_{i+gsize-1..i} = m[rc]_{j+gsize-1..j}$, i=0..128-gsize by gsize
- rc specifies address, msize, wsize
  - rc = base + msize/16 + wsize/16
  - vsize = msize/wsize

Summary

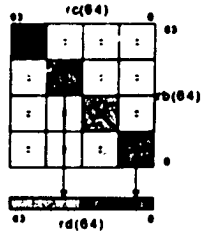
- Order-of-magnitude multiply performance increase
  - matrix multiply
  - convolve
- Wide switch: bit permutation
- Wide select: table lookup

BroadMX™ vs. MMX™

- Convolve Extract
  - 64 Multiplies
  - 56 Adds
  - 08 Extract w/round
- MMX Instructions
  - 16 MOV
  - 16 PMADDWD
  - 12 PADDD
  - 08 PSHW
  - 04 PSHR
  - 02 PACK
  - 58 total

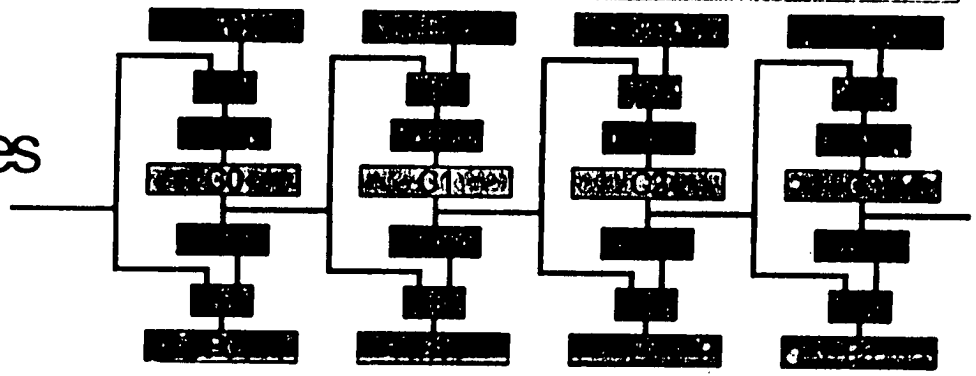


16

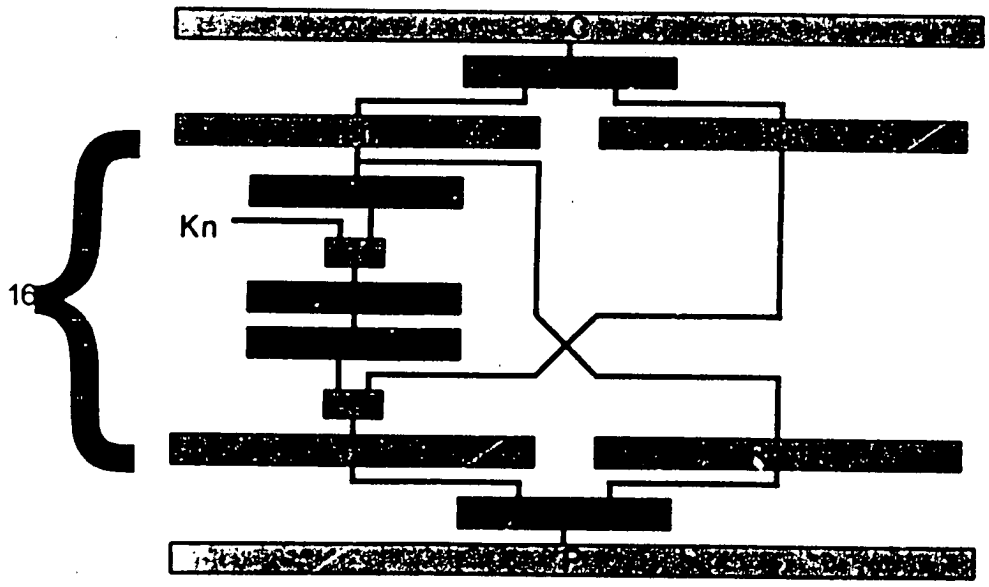


DES decryption

- CBC (Cypher Block Chaining) decrypt uses parallelism between blocks



- DES decrypt
  - E expansion
  - + key xor
  - S substitution
  - P permutation
  - + data xor



MicroUnity

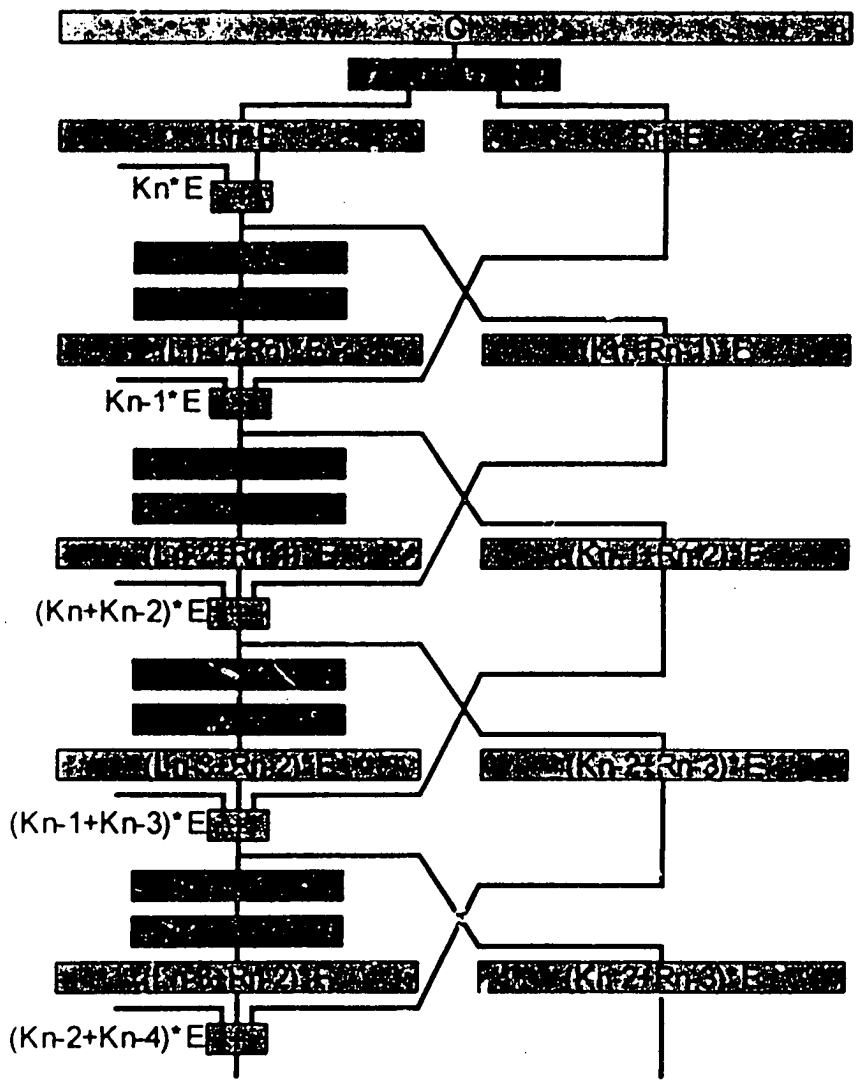
Software DES

- Optimizations
  - 2 blocks/register
  - 4 blocks at once
  - distribute E
  - combine + +
- Code

  K,+ L.128, G.XXX

  S W.TRANSLATE

  PE W.SWITCH
- Performance
  - 52 cycles/4 blocks
  - 985Mbps@200MHz
  - 10x per clock over fastest sw DES

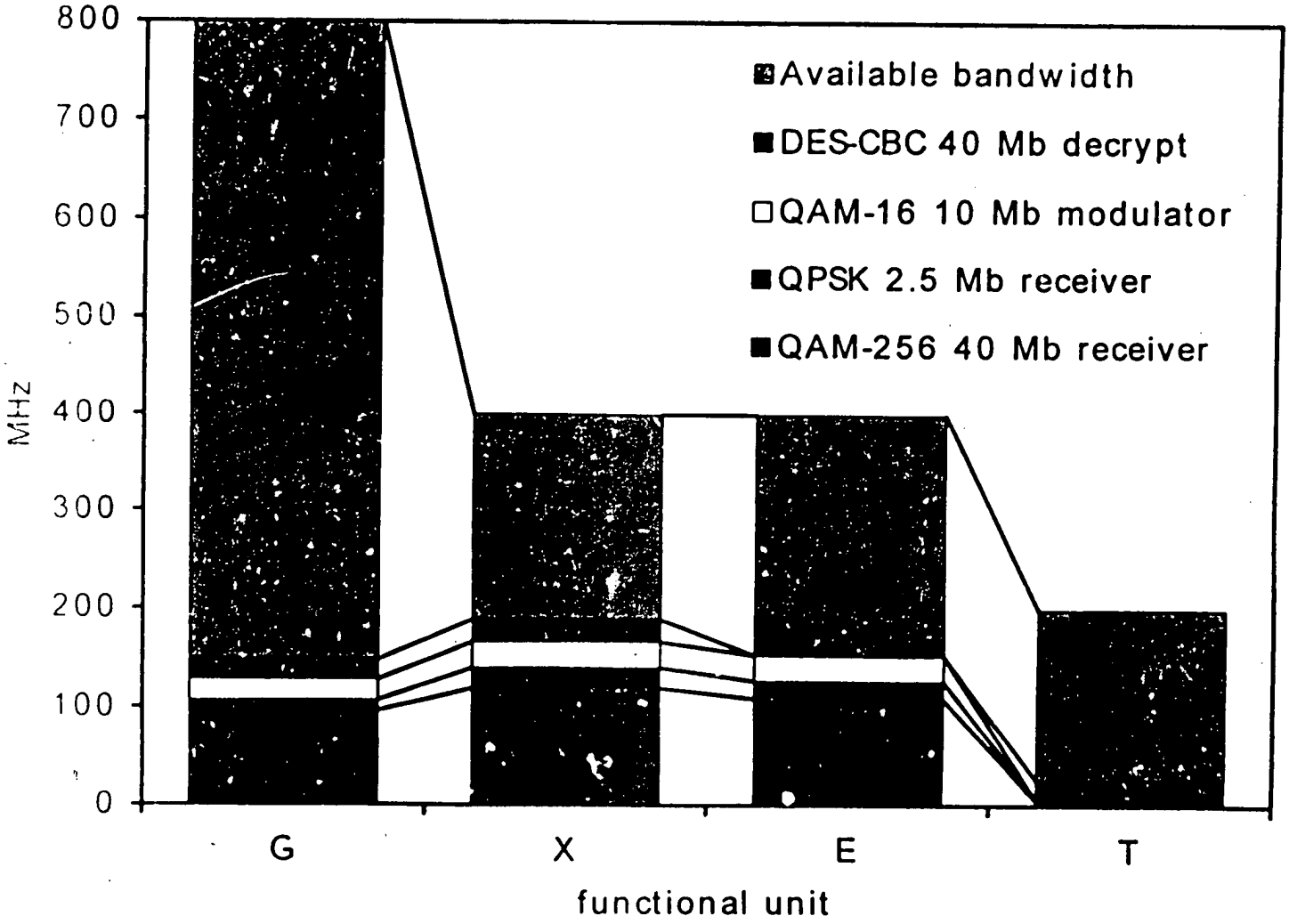


MicroUnity

Software DES

- DES standard at end of 20 year life
  - brute-force code-breaking
    - $10000 RSA DES Challenge
    - Electronic Frontier Foundation (EFF)
      - 56 hours to crack
      - $200k to design and build
  - FIPS standard expire this year
- Handles DES extensions
  - larger keys, bigger S-boxes
  - more rounds, larger blocks
  - soft S-boxes and P-boxes
- AES standard in development
  - 15 official candidates
  - new standard unpredictable

Instruction bandwidth for cable modem



MicroUnity

Software tools

- Compiler-based development tools
  - C, C++ compiler
    - intrinsic functions, function inlining
    - register allocation, code scheduling
    - future: automatic parallelisation
  - object-module tools
    - linker, libraries, debugger
- OS: RT microkernel, Linux
- DSP libraries
- Sophisticated tools
    - Mathematica: symbolic verification
    - GOPS: cross-development library